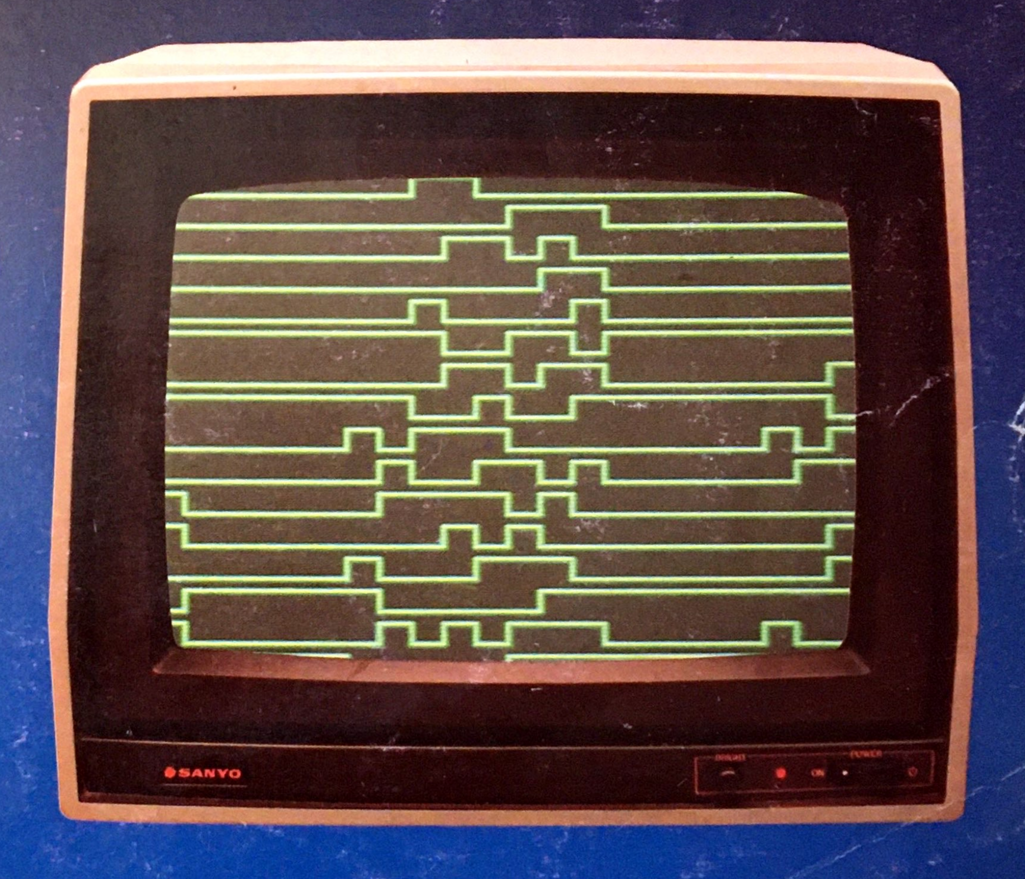
Sonderheft Nr. 88
Preis 28 DM
28 sfr., 210 öS

mc

MIKRO-COMPUTER

Schritt für Schritt



Alles zum Thema Mikroelektronik für den technisch Interessierten

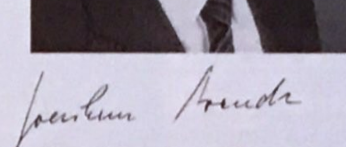
Mikrocomputer selbst gebaut

Vom Netzteil bis zur Software

Das Sonderheft zum NDR-Klein-Computer aus der Fernsehserie „Mikroelektronik" des NDR







Was wir mit diesem Heft wollen

Das hatten wir nicht erwartet, daß wir mit unserer Sendereihe Mikroelektronik so weitreichende Resonanz erzielen würden. Für die Schule konzipiert, das erste Mal im Frühjahr 1984 im Sendegebiet des NDR ausgestrahlt, zeigte sich, daß wir nicht nur bei den Schulen ins Schwarze getroffen haben, sondern vor allem auch bei einer großen Gruppe allgemeiner Zuschauer, die offenbar begierig nach kompetenten Informationen über die Mikrocomputer dürstete und dürstet. Diese Tatsache zeigte sich an über 40 000 Briefen und Zuschriften, Zustimmung und Kritik enthaltend.

Ein Problem wurde uns an einem Teil der Zuschriften klar. Unser Fernseh-Kurs im Medienverbund, bestehend aus 26 Sendungen, einem Buch, in dem alle Grundlagen geschildert sind, und vor allem aus einem Bausatzsystem, konzipiert für die Schule, ist ohne die Führung durch einen Lehrer für all die Zuschauer, die schon der Schule entwachsen sind, nicht immer einfach zu verstehen. All die Fragen, die ein Lehrer ganz schnell beantworten kann, die Koordinierung der einzelnen Quellen des Lernstoffes und die Hilfen beim Aufbau der Schaltungen, die ein Lehrer geben kann, fehlten den Nicht-Schülern unter den Zuschauern. Und auch mancher Lehrer, der den Kurs mit seinen Schülern zusammen durchmachen wollte, um im eigenen Unterricht mit den Schülern zu lernen, klagte, daß in dieser Situation doch noch Begleitmaterial fehle, das dem allein Lernenden als gesicherte Startbasis zum Eindringen in das System des NDR-Klein-Computers dienen kann.

Wir sind deshalb froh, daß wir jetzt zum Termin, an dem der Kurs durch die anderen Sendeanstalten der ARD zu wandern beginnt, ein Sonderheft vorlegen können, das für alle die geschrieben ist, die allein lernen wollen oder müssen. Aus all den Zuschriften und Reaktionen haben wir abgeleitet, was man benötigt, wenn man den NDR-Klein-Computer selbständig allein aufbauen möchte. Von den ersten theoretischen Grundlagen bis zum rechten Lötkolbenschwung, vom ersten Blinken bis zur Steuerung eines Roboters, vom ersten einfachen Befehl bis zum eigenen Programm, das Unterprogramme benutzt, können Sie jetzt alles im Heft nachlesen. Damit der Stoff nicht zu umfangreich wurde, haben wir uns auf die Schilderung des Z80-Systems beschränkt.

Wir hoffen damit, daß viele Verständnis- und Nachbauprobleme nicht mehr entstehen, und daß Ihr Weg in den Kurs Mikroelektronik von Stolpersteinen befreit ist. Wir wünschen uns, daß Ihnen dieses Heft über den Lernerfolg hinaus auch etwas Spaß macht. Schule, das ist ja manchmal mit trockenem Stil und gelehrtem Dozieren verbunden. Wir haben uns bemüht, zu zeigen, daß das bei dem Thema Mikroelektronik, das uns genauso wie Sie fasziniert, nicht so sein muß.

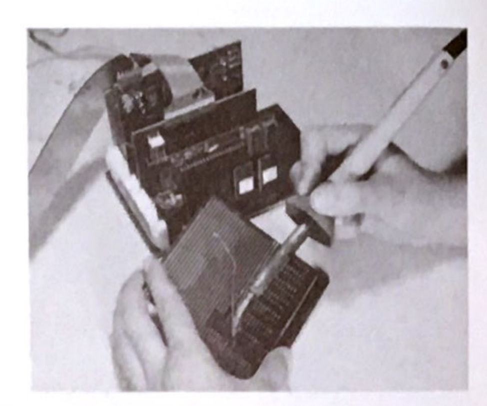
Was Sie brauchen

Wer sich den Computern von der praktischen Seite her nähern will, der benötigt eine kleine **Grundausstattung an Handwerkszeug.** Es ist wirklich wenig, nämlich nur: Ein Feinlötkolben, Bleistift-Form, maximal 30 W; ein Seitenschneider (zum Abtrennen der Bauelementeanschlüsse und zum Ablängen von Schaltdraht); ein Schraubendreher (Elektronik-Ausführung mit isolierendem Kunststoff-Griff); eine Spitz-Zange (für den Zugriff auf Bauteile in beengter Umgebung); eine Pinzette (kräftige Ausführung). Zum Löten wird Elektroniklot benutzt, das im Inneren eine geeignete Flußmittelseele besitzt. Die Stärke des Lötzinn-Drahtes sollte etwa 1 mm betragen. Mit diesem **Werkzeug** kommt man durch das ganze Heft, denn die Bausätze selbst sind einfach aufzubauen. Allerdings benötigt man zum Prüfen und Messen ein **Universal-Meßgerät,** mit dem man Spannungen, Ströme und Widerstände messen kann. Die Ausführung sollte ein Vielfach-Meßinstrument mit 20 kΩ pro Volt Innenwiderstand sein. Ein Zeigerinstrument mit Drehspulmeßwerk ist absolut ausreichend. Elektronische Multimeter sind für die Überprüfung der Schaltungen aus diesem Heft nicht geeignet, da sie in manchen Situationen keine brauchbare Anzeige liefern.

Beim **Aufbau** des NDR-Klein-Computers – und beim Betrieb – kommt man völlig **ohne Oszilloskop** aus.

Wenn man aber Zugang zu einem solchen Gerät hat, dann kann man viele beeindruckende Messungen durchführen, wenn es sich um ein Gerät mit 10 MHz Grenzfrequenz handelt, das extern triggerbar ist. Für Arbeitsgruppen und Kurse empfiehlt sich ein Zweistrahl-Oszilloskop, weil dann Signale in ihrer gegenseitigen Beziehung dargestellt werden können. Nochmals gesagt, für den Privatmann ist die Anschaffung eines Oszilloskopes allein zum Aufbau und Betrieb des NDR-Klein-Computers nicht notwendig.

Der NDR-Klein-Computer



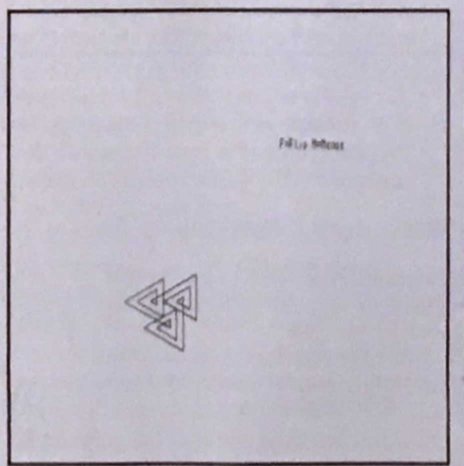
Inhalt



In diesem Heft wird der NDR-Klein-Computer Schritt für Schritt aufgebaut. Erste Experimente kann man schon mit geringem Aufwand durchführen. Nach den letzten Schritten in Kapitel 11 besitzt man einen selbstgebauten Computer, der es in sich hat.

Man kann diesen Computer nämlich systematisch erweitern. Der NDR-Klein-Computer ist ein ganzes System, das aufrüstbar ist mit Floppy-Laufwerken oder umrüstbar zu einem 16-Bit-Computer.

In seiner Grundversion kann er besonders gut zeichnen, das ist an kleinen Beispielen bewiesen, die Sie überall im Heft verstreut finden. Diese Beispiele verstehen Sie ab Kapitel 12.



Vorkenntnisse benötigen Sie im Prinzip keine. Im Prinzip heißt hier, daß Sie schon wissen, was es mit Strom, Spannung und einfacher Elektrik so auf sich hat. Auch sollten Sie einen einfachen Schaltplan, etwa mit Batterie, Glühlampe und Schalter, einen Stromkreis also, schon einmal gesehen haben. Alles andere wird im Heft erklärt.

Für die **ersten beiden Kapitel** in diesem Heft benötigen Sie an Baumaterialien: 2 Transistoren BC 107, 2 Widerstände mit je 1 kΩ, eine Glühlampe, 6 V/50 mA, mit Lötanschlüssen, eine kleine Lochrasterplatine, auf der Sie die Experimente aufbauen. Das gibt es alles im Elektronikladen um die Ecke. Eine Batterie 4,5 V (Flachbatterie) und etwas Schaltdraht.

Für das Kapitel **„Vom Schaltplan zum Gerät“** bestellen Sie sich den Bausatz „Prüfstift“, denn nur mit diesem Prüfstift können Sie später die Messungen durchführen, die geschildert werden. Ein anderer Prüfstift, aus dem allgemeinen Angebot, ist nicht brauchbar. Außerdem ist es billig und einfach, die ersten Lötübungen an unserem Prüfstift-Bausatz durchzuführen. Fürs erste kann er mit einer frischen 4,5-V-Flachbatterie betrieben werden. Die **Spannungsversorgung** wird wiederum mit einem Bausatz (POW5V) aufgebaut, in dem aber der Trafo mit dem Netzstecker und dem VDE-mäßigen Gehäuse nicht enthalten ist. Den muß man sich im örtlichen Elektronikhandel besorgen oder bei einem Versand bestellen. Für die ersten Experimente wird hier das Meßgerät benötigt. Arbeitsgruppen und Schulen setzen ein Oszilloskop ein. Zum **ersten Ausbau** und für die **weiteren Kapitel** benötigt man den Bausatz SBC2 (Single Board Computer). Mit dem Werkzeug, dem Prüfstift und dem Vielfachmeßinstrument kann er sicher aufgebaut werden. Zusätzlich benötigt man nur etwas Schaltdraht, um die Stromversorgung mit POW5V sicherzustellen. Für die **Folge 6** (Nichtstu-Befehl) werden zu Trafo, SBC2 und POW5V noch 3 DIL-Fassungen, 24polig, für Experimente benötigt. Mit dem Prüfstift oder einem Oszilloskop werden die Messungen durchgeführt. Zu **Folge 7** (dem Speicher auf der Spur) wird kein neues Material benötigt. Zu **Folge 8** (EPROM macht Musik) werden der Bausatz Musik (oder die Teile dafür) und die EPROMs MUO, MUM und RWT benötigt, die Musik ohne RAM, Musik mit RAM erzeugen und einen Speichertest enthalten. Zu **Folge 9** (Alarmstufe Rot) gibt es einen Bausatz für die Ampelanlage. Voraussetzung ist dazu noch ein Bausatz namens IOE, der dabei hilft, Signale in den und aus dem Computer heraus zu bringen. Er ist für alles weitere grundlegend. Zusätzlich ist zu diesem Zeitpunkt die Beschaffung der BUS-Platine empfehlenswert. An Software in EPROM wird das Programm AMPEL benötigt. Zu **Folge 10** (Roboter steuern) werden ein Fischertechnik-Baukasten „Computing“ benötigt sowie der Bausatz zur Roboter-Steuerung und ein entsprechendes EPROM. Zu **Folge 11, 12, 13** werden eine Tastatur nach DIN im Gehäuse, der Bausatz GDP64, der Bausatz KEY und ein Video-Monitor benötigt. An Software kommt der Monitor, das RDK-Grundprogramm, dazu.

Zu **Folge 14** benötigt man einen Kassettenrecorder und den Bausatz CAS sowie das EPROM SKOP. Zu **Folge 15** benötigt man die universelle Experimentierplatine aus dem Kapitel „Roboter steuern“, einen Gleichstrom-Motor und etwas bastlerisches Geschick.

Je nach Geschmack benötigen Sie nur noch die EPROMs BASIC oder GOSI für die **restlichen Kapitel** dieses Heftes. Außerdem vielleicht die vielen Erweiterungs-Bausteine, die es mittlerweile für den Computer gibt. Zum Beispiel mehr Speicher. Unbedingt benötigen Sie später das Zilog-Z80-Datenbuch (in Englisch, Vertrieb Kontron, München); nützlich ist für alle, die Englisch können, das Buch „Z80 Assembly Language Programming“ von Lance A. Leventhal, erhältlich beim tewi-Verlag München.

Bestellen Sie aber **nicht alles gleich aufeinmal,** das Material wird erst nach und nach benötigt, verteilen Sie die Einkäufe über die Kursdauer. Dann werden Sie auch schon wissen, ob Sie ein einzelnes Kapitel ganz genau durcharbeiten wollen oder nicht.

Der Wegweiser durch die Kursmaterialien

Das Fernsehen sollte das Leitmedium sein. Wahrscheinlich wurden Sie durch das Fernsehen überhaupt auf den Kurs aufmerksam gemacht. Das Fernsehen reißt an, greift die Themen auf und führt Ihnen vor, wie Fachleute und Laien, vom Moderator geführt, mit dem NDR-Klein-Computer umgehen, an ihm lernen und ihn benutzen. Das Fernsehen zeigt, was und wie man es lernen kann.

Das Sonderheft enthält, didaktisch für den Anfänger und allein Lernenden gut aufbereitet, alles das, was man überhaupt benötigt, um die ersten Schritte in die Mikroelektronik anhand des Mikrocomputers zu tun. Es ergänzt und begleitet das Fernsehen und ist die Grundlage des Lernstoffes, weil alles, was im Fernsehen geschildert wurde, hier schwarz auf weiß zusammengefaßt ist, damit Sie es immer präsent haben, sobald Sie etwas wissen wollen. Darüber hinaus sind viele Dinge im Heft geschildert, die notwendigerweise im Fernsehen zu kurz kommen müssen, weil sie nur auf Papier richtig präsentabel sind. Längere Mikrocomputerprogramme zum Beispiel. Notfalls kann man sogar mit dem Sonderheft allein beginnen, sei es, daß man die Sendetermine verpaßt hat, sei es, daß man nicht warten will, bis das entsprechende dritte Programm die Reihe Mikroelektronik ausstrahlt.

Das Buch „Mikrocomputer selbst gebaut und programmiert" ist der technische Hintergrund Ihres Weges in die Mikroelektronik. Sie werden feststellen, wenn Sie nur ein bißchen angebissen haben, daß man dann nie genug wissen kann. Im Buch sind wiederum alle Schaltungen geschildert, aber so detailliert und mit exakter Beschreibung der Funktion, daß auch Vollbluttechniker zufrieden sein werden. Sie werden feststellen, daß Sie in Stufen Ihre Kenntnisse erweitern werden. Fernsehen und Sonderheft und dann das Buch.

Die Bausätze und die **EPROMs** sind das Neue an diesem Kurs. Der Einstieg in die Mikroelektronik von der technischen Seite her, wie es hier im Kurs geschehen soll, kann nicht ohne selbstgefertigte Elektronik erfolgreich sein. Löten lernen heißt hier nicht Basteln lernen, sondern technisches Verständnis erwerben. Das Ganze ist preiswert. Die Firma Graf Elektronik in Kempten hat das technische und pädagogische Konzept von Rolf-Dieter Klein (daraus entstand der Name NDR-Klein-Computer) umgesetzt in ein modulares System, das einmalig ist. Genau die Idee von Rolf-Dieter Klein, daß man ein nach oben (und auch nach den Seiten) offenes Hardware-Konzept haben müsse, wenn man den Anfänger vom ersten Schritt bis in die Welt der Mikrocomputerprofis führen will, ist von der Firma Graf Elektronik mit Sorgfalt in eine Reihe von handelsreifen Bausätzen und Platinen umgesetzt worden. In Lizenz werden die Bausätze und Platinen auch von der Firma Elektronikladen Detmold vertrieben. Die Speicherbausteine, in welchen die Grundprogramme (und vieles mehr) enthalten sind, die Ihren Computer zunächst in Schwung bringen, liegen den Bausätzen bei oder können im Handel oder beim Franzis-Verlag, Abteilung Softwareservice, in Form von EPROMs bezogen werden.

Andere Materialien. Das Kurssystem Mikroelektronik ist so erfolgreich, daß über den Kreis der Autoren hinaus sich viele Personen mit der Didaktik oder dem Inhalt des Kurses beschäftigt haben und auch Materialien dazu entwickelt haben, seien es Programme, seien es Papiere oder Bücher für den Lehrer oder gar ganze Begleitkurse, wie sie in Volkshochschulen und Handwerkskammern schon stattgefunden haben. Einzelheiten erfahren Sie über die An- oder Absagen zu den Sendungen.

Impressum

1983, Franzis-Verlag GmbH, Karlstraße 37–41, D-8000 München 2.

Bearbeitet von der Redaktion der Zeitschrift mc. Für den Text verantwortlich: Dipl.-Math. Ulrich Rohde.

ISSN 0722-0022. Druck: Franzis-Druck GmbH, München. Printed in Germany. Imprimé en Allemagne.
ZV-Art.-Nr. 88041 · F/ZV/784/1015/10'

Hans Hehl

Von den ersten theoretischen und praktischen Grundlagen

Material zur Einstimmung, Mikroelektronik, Folge 0

Im Jahr 1941 wurde in Deutschland die erste programmgesteuerte, elektrische Rechenanlage in Betrieb genommen. Das Gerät bestand aus 2600 höchst sinnvoll verschalteten Relais. Erdacht hatte diesen ersten Computer der Welt, der auch wirklich vollständig funktionierte, ein Mann namens Konrad Zuse, der damit die langwierigen und auch fehlerträchtigen Berechnungen im Bauingenieurswesen und auch anderswo automatisieren und damit sicherer und schneller durchführbar machen wollte. Der Computer trug den Namen Z3. Er ist im Deutschen Museum in München in Lebensgröße zu besichtigen.

Eingeschaltet, ausgeschaltet

Die Z3 von Konrad Zuse war ein erstaunliches Gerät. Denn sie konnte neben den Grundrechenarten zum Beispiel auch Wurzeln ziehen. Das ist deshalb so erstaunlich, weil die Z3 dazu (natürlich in elementarer Form) alle Merkmale programmgesteuerter Universalrechenmaschinen aufweisen mußte, wie sie auch heute noch gültig sind. Erfunden und in die Maschine eingebaut hat das alles im wesentlichen ein einziger Mann, eben Konrad Zuse, der dafür in den Jahren nach 1970 auch vielfältig geehrt wurde.
Eine seiner wichtigen Ideen beim Bau der Z3 war die Verwendung eines Zahlensystemes, das besser an Maschinen angepaßt ist als unser gewöhnliches Zehnersystem. Intern rechnete die Z3 mit den sogenannten Dualzahlen. Weshalb das so gut war, soll gleich erklärt werden: Es hängt mit der Verwendung von Relais zusammen.
Ein Relais ist ja nichts weiter als ein elektrischer Schalter, der mit elektrischem Strom ein- und ausgeschaltet werden kann. Bei einem Relais macht also der Strom das, was man bei einer Taschenlampe mit dem Daumen von Hand machen muß: Man schaltet den Schalter der Lampe ein oder aus. Entsprechend wird die Lampe leuchten oder nicht. Eine Verbindung zu den Zahlen kann man schlagen, wenn man verabredet, daß zum Beispiel der Zustand der Taschenlampe Null sein soll, wenn sie ausgeschaltet ist und Eins, wenn sie eingeschaltet ist. So merkwürdig künstlich und willkürlich so eine Verabredung zunächst erscheint, die ganze Computerindustrie ist in gewissem Sinn darauf aufgebaut.
Ein Relais, eine Taschenlampe, überhaupt ein physikalisches Gerät, das zwei Zustände annehmen kann, von welchen man verabreden kann, daß der eine Zustand Null, der andere Eins bedeuten soll, das sind Beispiele für die Realisierung einer sogenannten binären Variablen.
Einer Taschenlampe sieht man nicht an, mit welcher Spannung die Glühbirne betrieben wird. Allerdings weiß man, daß sicher nicht so hohe Spannungen wie z. B. 220 V verwendet werden. Genauso sind die Spannungswerte (Pegel) bei einer elektrisch dargestellten binären Variablen (ein- oder ausgeschaltet) prinzipiell nicht vorgegeben. Sie hängen jeweils von der technischen Konzeption des Gerätes ab. Dem Binärwert 0 kann man zum Beispiel die Spannung 0 V ebenso zuordnen wie die Spannung −12 V. Allgemein spricht man von einem L-Pegel (Low), wenn der Pegelwert näher bei „minus unendlich" liegt und von einem H-Pegel (High), wenn der Pegelwert näher bei „plus unendlich" liegt. In der Praxis wird meist dem L-Pegel der Wert 0 der binären Variable und dem H-Pegel der Wert 1 zugeordnet (positive Logik).

Dualzahlen binär dargestellt

Wenn man begreifen will, wie Zahlen in einem Computer dargestellt werden, dann ist es zum Beispiel günstig, sich den Kilometerzähler eines Autos vorzustellen. Es sei ein Modell, das keine Hundert-Meter-Einteilung besitzt. Dann wird dort beim Fahren das Anzeigerad für die einzelnen Kilometer, also das Rad ganz rechts, von einer mit den Auto-Rädern verbundenen Welle gedreht und zeigt nacheinander 0, 1, 2, 3, 4,... Auf diesem Anzeigerad (und auf den anderen weiter links auch) befinden sich zehn Ziffern, die der Reihe nach gezeigt werden. Immer dann, wenn das Einerrad eine Umdrehung vollendet, also beim Umschalten von 9 auf 0, wird das benachbarte weiter links befindliche Rad um eine Ziffer weitergedreht. Stand es vorher auf 0, dann steht es nach einer solchen Situation auf 1. Dazu besitzt das Einerrad einen Mitnehmer, der das Zehnerrad dann mitnimmt. Also nach zehn gefahrenen Kilometern steht tatsächlich auch 10 auf dem Kilometerzähler. Das Zehnerrad zählt also mit, wie oft das Einerrad sich gedreht hat und merkt so an, wievielmal zehn Kilometer zurückgelegt wurden. Das Zehnerrad selbst besitzt ebenfalls einen Mitnehmer, der bei Vollendung einer Umdrehung das benachbarte Hunderterrad um eins weiterdreht. Und dieses Hunderterrad wiederum kann das Tausenderrad mitnehmen, was selbst wieder das Zehntausenderrad mitnimmt. Und so weiter.

Daß die Anzeigeräder jeweils 10 Ziffern tragen, das rührt von unserer Gewohnheit her, im Zehnersystem zu rechnen. Eine ziemlich merkwürdige, aber durchaus mögliche Konstruktion eines solchen Kilometerzählers könnte darin bestehen, daß man auf die Anzeigeräder nur auf der einen Seite des Umfanges 0 und auf der anderen Hälfte 1 anschreibt und den Mitnahmemechanismus so gestaltet, daß beim Drehen von 1 auf 0 das weiter links befindliche Rad um eine halbe Umdrehung weiter gedreht wird. Das Einerrad zählt dann von 0 bis 1. Links daneben befindet sich das Zweierrad, das um Eins weiter gedreht wird, wenn das Einerrad einmal ganz herum kommt und dabei der zweite Kilometer abgefahren wird. Nach zwei gefahrenen Kilometern steht dann 10 auf diesem merkwürdigen Zähler. Auch das Zweierrad dreht beim Übergang von 1 auf 0 ein weiter links befindliches Rad. Es sind hier einfach einmal für einen vierstelligen Zähler mit der verrückten Zweiereinteilung die Anzeigestellungen und die gefahrenen Kilometer aufgezählt:

0000	=	0
0001	=	1
0010	=	2
0011	=	3
0100	=	4
0101	=	5
0110	=	6
0111	=	7
1000	=	8
1001	=	9
1010	=	10
1011	=	11
1100	=	12
1101	=	13
1110	=	14
1111	=	15

An dieser Aufstellung kann man, wenn man scharf hinschaut, erkennen, daß in einer Kolonne von oben nach unten (links vom Gleichheitszeichen) immer nur die Ziffern 0 und 1 auftauchen. Man könnte also für jede der vier Stellen eine binäre Variable hernehmen und diese vier Variablen dann jeweils so schalten, wie es das Muster aus Nullen und Einsen verlangt, das gerade auf dem Kilometerzähler erscheint. Zum Beispiel leuchtet bei vier nebeneinanderliegenden Taschenlampen genau die ganz rechts außen liegende. Dann kann man sagen, daß damit die Zahl Eins binär dargestellt ist. Wenn alle vier Lampen eingeschaltet sind, dann ist damit die Zahl 15 binär dargestellt. Jeder Kombination von „Ein" und „Aus" entspricht also ganz natürlich eine Zahl.

Mathematiker nennen Gebilde, wie sie in der linken Spalte auftauchen, Dualzahlen, wenn sie betonen wollen, daß sie in einem System arbeiten, das nur die Ziffern 0 und 1 benutzt. Wie eben gesagt, kann man also die Dualzahlen zum Beispiel mit einer geeigneten Anzahl von Taschenlampen binär darstellen. Zuse benutzte in seiner Z3 Relais, um damit Dualzahlen binär in seiner Maschine darzustellen.

Noch etwas Theorie

Ein Zahlensystem mit den beiden Ziffern 0 und 1 unterscheidet sich von unserem gewohnten Zehnersystem durch einen wesentlich kleineren Abstand der Stellenwerte. Was besagt dies aber? Grundsätzlich können wir je nach Art der Anordnung von Zeichen für Zahlen Additionssysteme und Positionssysteme unterscheiden.

Ein Additionssystem ist zum Beispiel das römische Zahlensystem. In ihm wird das Jahr 1768 als MDCCLXVIII dargestellt. Die eigentliche Zahl ergibt sich durch Addition der einzelnen Zahlzeichen. Das heutzutage benützte Positionssystem (auch Stellenwertsystem genannt) wertet dagegen die Stellung des Zahlzeichens mit aus.

Tabelle 1. Dezimalzahl und Einschaltkombination

Dezimalzahl	Kombination
1	0 0 0 0 0 0 0 1
2	0 0 0 0 0 0 1 0
3	0 0 0 0 0 0 1 1
4	0 0 0 0 0 1 0 0
5	0 0 0 0 0 1 0 1
6	0 0 0 0 0 1 1 0
7	0 0 0 0 0 1 1 1
8	0 0 0 0 1 0 0 0
9	0 0 0 0 1 0 0 1
10	0 0 0 0 1 0 1 0
.	.
31	0 0 0 1 1 1 1 1
.	.
255	1 1 1 1 1 1 1 1

Ein Beispiel soll dies verdeutlichen: In dem römischen Zeichen III für die Zahl Drei hat jede der drei Ziffern den Zahlwert 1 und die Zahl ergibt sich durch die Addition der drei einzelnen Zahlenwerte.

Im dekadischen Positionssystem ergibt die Zeichenanordnung 111 die Zahl Einhundertelf. Alle Zeichen haben den gleichen Zahlwert 1, aber einen unterschiedlichen Stellenwert. Der niedrigste Stellenwert (Einer) steht ganz rechts, dann folgen Zehner und Hunderter. Jeder Stellenwert beträgt 1/10 des links von ihm stehenden.

Beim Binärsystem sind die Stellenwerte die Potenzen der Zahl (Basis) 2, also die Zahlen 1, 2, 4, 8, 16, 32, 64,... usw.

Acht Taschenlampen oder ein Byte?

Wir verwenden nun acht Taschenlampen, die wir in eine Reihe legen und beliebig ein- bzw. ausschalten. Damit ergeben sich $2^8 = 256$ Einschaltkombinationen. Wenn die Zuordnung dieser Kombinationen zu den Zahlen 0 bis 255 mit der Rechenregel „Dezimalzahl ergibt sich durch Aufsummieren der Zweierpotenzen" durchgeführt wird, dann ergibt sich das Schema wie beim Kilometerzähler. In *Tabelle 1* sind einige Dezimalzahlen und die entsprechenden Kombinationen der Werte 0 und 1 aufgeführt (eingeschaltet = 1, ausgeschaltet = 0), die sich beim Zählen wie vorhin ergeben würden.
Die Zuordnung kann nun überprüft werden. Die Kombination 0 0 0 1 1 1 1 1 ergibt als Summe der Zweierpotenzen die Zahl 31.

$$0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$
$$0 + 0 + 0 + 17 + 8 + 4 + 2 + 1$$

Umgekehrt kann aus einer Dezimalzahl zwischen 0 und 255 die zugehörige Dualzahl ermittelt werden, indem fortlaufend die Zweierpotenzen, beginnend bei

Tabelle 2: Gegenüberstellung der drei Stellenwertsysteme

Dezimal	Sedezimal	Dual
0	0	0000 0000
1	1	0000 0001
2	2	0000 0010
3	3	0000 0011
.	.	.
8	8	0000 1000
9	9	0000 1001
10	A	0000 1010
11	B	0000 1001
12	C	0000 1100
13	D	0000 1101
14	E	0000 1110
15	F	0000 1111
16	10	0001 0000
17	11	0001 0001
.	.	.
94	5E	0101 1110
.	.	.
171	AB	1010 1011
.	.	.
255	FF	1111 1111

2^7, von der Zahl bzw. vom übrig bleibenden Rest abgezogen werden. Würde die Differenz negativ, so wird eine Null aufgeschrieben und die nächst kleinere Zweierpotenz verwendet. Ist die Differenz positiv, so wird eine 1 aufgeschrieben und mit dem Rest weiter gearbeitet. Probieren wir dies mit der Zahl 18 aus.

18 − 128 = ?	geht nicht,	also 0
18 − 64 = ?	geht nicht,	also 0
18 − 32 = ?	geht nicht,	also 0
18 − 16 = 2	geht	, also 1
2 − 8 = ?	geht nicht,	also 0
2 − 4 = ?	geht nicht,	also 0
2 − 2 = 0	geht	, also 1
0 − 1 = ?	geht nicht,	also 0

Die Kombination für die Zahl 18 lautet also von oben nach unten: 0 0 0 1 0 0 1 0.

Eine solche Kombination von Nullen und Einsen nennt man „Byte". Dieses Byte besteht aus acht Bits. Ein Bit, abgeleitet von „binary digit", ist die kleinste Darstellungseinheit für Binärdaten. Ein Bit repräsentiert eine Binärstelle in einem Byte.

Für Umwandlungsübungen seien noch einige Beispiele angegeben.

85 = 0 1 0 1 0 1 0 1
138 = 1 0 0 0 1 0 1 0
255 = 1 1 1 1 1 1 1 1

Vom Relais zum Transistor

Genug der Mathematik, nun sei diskutiert, wie das Ein- und Ausschalten der Glühbirnchen unserer acht Taschenlampen automatisiert werden kann. 1941 verwendete K. Zuse dazu Relais. Aber so ein Relais kann nicht hunderte von Schaltvorgängen pro Sekunde durchführen, die Kontakte sind dazu zu träge. Es ist schon eigenartig, daß nur 7 Jahre später, also 1948 ein Ersatz für das langsame Relais entdeckt wurde. Die Amerikaner Bardeen, Brattain und Shockley entdeckten den Transistoreffekt an einem Germaniumkristall und erhielten dafür 1956 den Nobelpreis für Physik.

Mit dem Transistor stand ein Bauteil zur Verfügung, das keine mechanischen Teile enthält, sehr schnell schalten kann und weniger Strom als der Elektromagnet eines Relais benötigt. *Bild 1* zeigt den Schaltplan eines Transistorschalters.

So eine Schaltung wird zum Beispiel benötigt, wenn ein Computer Lampen, Motoren usw. schalten soll, um also eine Verbindung zur Außenwelt zu schaffen. Bevor wir die Teile der Schaltung näher betrachten, müssen wir uns jedoch dem Problem der Stromrichtung zuwenden.

Man hatte vor der Zeit der Elektronenröhren und der Halbleiter einfach die Stromrichtung vom Pluspol der Spannungsquelle über den Verbraucher zum Minuspol festgesetzt (technische Stromrichtung). Die normalen Träger der Elektrizität, die Elektronen, fließen aber vom Minuspol über den Verbraucher zum Pluspol, wie man erst später entdeckte. Wir verwenden hier diese Elektronenflußrichtung.

Wichtigster Teil der Treiberschaltung nach Bild 1 ist der Transistor (BC-107). Er besteht im Inneren aus drei Halbleiterschichten mit wechselnder Leitfähigkeit. Halbleiter, zum Beispiel Silizium oder Germanium, leiten den Strom schlechter als Metalle. Werden geringste Mengen eines anderen Metalles (z. B. Antimon) hinzugefügt, verändert sich die Leitfähigkeit der Schicht erheblich. Verfolgen wir nun den Elektronenfluß durch den Transistor. Vom Minuspol

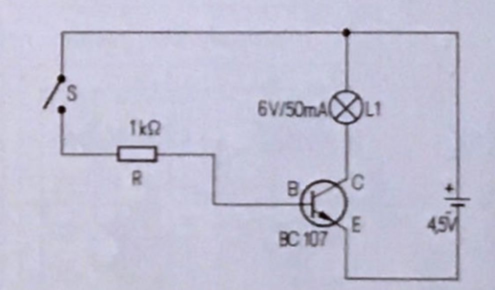


Bild 1. Treiberschaltung mit einem Transistor

der Batterie fließen die Elektronen zum Emitter E (durch eine Pfeilspitze gekennzeichnet, die aus dem Kreissymbol heraus zeigt, npn-Typ). Wieviel Elektronen nun zum Collektor C (bzw. Kollektor) und damit durch die Lampe L1 fließen können, hängt vom Stromfluß am Steuereingang (Basis) B ab, der vom Emitter über Basis, Widerstand R, geschlossenen Schalter S zum Pluspol der Batterie fließt. Ein geringer Emitter-Basis-Strom bewirkt einen großen Emitter-Kollektor-Strom, man spricht von einer „Stromverstärkung". Großer Strom bedeutet aber nach dem Ohmschen Gesetz einen kleinen Widerstand der Emitter-Kollektor-Strecke, der Transistor „schaltet durch". Die Stromverstärkung beträgt einige „Hundertfache".

Nach der Theorie zur Praxis. Verwendet werden kann der Transistor BC-107 oder ein vergleichbarer npn-Typ (wie z. B. BC-109 und andere). Die Kennzeichnung der drei Anschlußdrähte des Transistors ergibt sich aus *Bild 2*. Der Draht dicht neben dem Gehäusevorsprung ist der Emitter, wobei man den Transistor von unten, also von der Anschlußdrahtseite her anschaut.

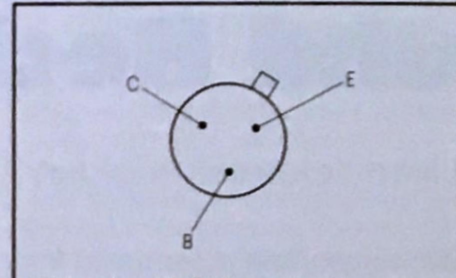


Bild 2. Anschlußschema des Transistors BC-107

Achtung: wird der Basis-Widerstand R überbrückt, liegt die Spannung der Batterie voll an Emitter und Basis an, ein großer Strom fließt und eine Zuleitung im Transistor schmilzt durch.

Löten: Übung macht den Meister

Wenn Sie die im Heft angegebenen Schaltungen aufbauen wollen, dann verwenden Sie bitte einen kleinen Lötkolben mit etwa 20 W Leistung und eine feine Dauerlötspitze, die nicht verzundert. Reine Kupferspitzen sind weniger geeignet. Die Spitze reinigt man vor jedem Lötvorgang mit einem feuchten Spezialschwämmchen oder mit einem Baumwollappen. Als Lot wird ein 1 mm dünner Lötdraht verwendet, der im Inneren ein Flußmittel auf Harzbasis enthält. Säurehaltige Flußmittel wie Lötfette oder sogar Salzsäure dürfen auf gar keinen Fall verwendet werden, da die Säurereste eine Korrosion der Leiterbahnen und Bauteile bewirken. Zum Löten erwärmen Sie mit der Lötkolbenspitze solange die zu verbindenden Teile, bis das gleichzeitig an die Teile gehaltene Lot schmilzt und die Teile überzieht. Die Lötstelle darf bis zum Erkalten nicht bewegt werden. Eine gute Lötstelle verbindet die Bauteile mit nur wenig Lötzinn, das eine hellglänzende Oberfläche besitzt. Üben Sie dies nicht mit Ihren Bausatzplatinen, sondern an versilberten Schaltdrahtresten oder Kupferlitze. Alles Handwerkszeug und das Leitzinn sollten Sie im Elektronikfachhandel kaufen, damit Sie sicher sind, daß Ihre Arbeitsmittel auch geeignet sind.

Hans Hehl

Vom Verknüpfen und Rechnen

Mikroelektronik, Folgen 1 und 2

Sie wissen nun schon, wie Informationen (z. B. Zahlen) in einem Rechner dargestellt werden: Durch Aneinanderreihen der Zahlen Null und Eins nach bestimmten Vorschriften. Wie kann aber der Rechner mit diesen Zahlenfolgen rechnen? Es müssen also einerseits Verknüpfungen zwischen diesen Zahlengruppen nach bestimmten Regeln durchgeführt werden und andererseits auch entsprechende elektronische Schalteinrichtungen vorhanden sein.

Treiber mal zwei

Wir benötigen zu Versuchszwecken wieder unsere Treiberschaltung aus dem ersten Abschnitt, Bild 1. Die Lampe leuchtete, wenn der Schalter geschlossen wurde.
Diese Treiberschaltung sei jetzt erweitert. Anstelle des Schalters wurde nochmals die gleiche Treiberschaltung eingesetzt. *Bild 1* zeigt die neue Schaltung. Nach dem Anschließen der Batterie leuchtet Lampe L1. Wird aber der Schalter S geschlossen, so erlischt Lampe L1 und L2 leuchtet. Warum?
Lampe L1 leuchtet zunächst, da ein kleiner Steuerstrom vom Transistor T1 über Widerstand und Glühfaden von L2 fließen kann, ohne daß L2 leuchtet. Da der Schalter offen ist, kann kein Steuerstrom bei Transistor T2 und damit auch kein Strom von dessen Emitter zum Kollektor fließen. Der Transistor T2 besitzt in diesem Zustand keinen Einfluß auf die Schaltung. Schließen wir aber den Schalter, so bewirkt der entstehende Steuerstrom einen großen Stromfluß durch T2 und die Lampe L2. Wir könnten auch Emitter und Kollektor von T2 mit einem Draht überbrücken, denn großer Stromfluß bedeutet kleinen Widerstand (Spannung konstant). Jetzt fließt kein Steuerstrom mehr durch T1, weil die Spannung am Kollektor von T2 fast ganz auf Null abgesunken ist. L1 erlischt deshalb.

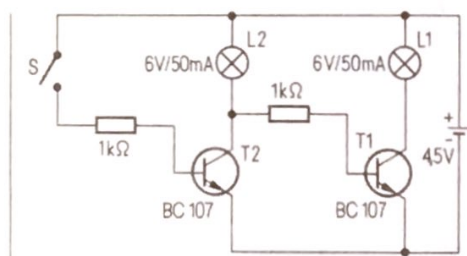


Bild 1. NICHT-Glied: Doppelte Treiberschaltung

Wir bezeichnen nun den Schalter als Eingang, dessen offenen Schalterzustand mit der Zahl 0 und den geschlossenen mit 1. Die Lampe L1 wird zum Ausgang erklärt. Den Leuchtzustand kennzeichnen wir mit der Zahl 1. Dann erhalten wir folgenden Zusammenhang:

Schalter	Lampe L1
0	1
1	0

Das Eingangssignal erscheint am Ausgang invertiert, also genau umgekehrt. So eine invertierende Schaltung wird als NICHT-Glied bezeichnet. *Bild 2*zeigt das Schaltsymbol der invertierenden Schaltung.

Nicht nur NICHT

Ein Gesichtspunkt ist besonders interessant. Man kann das Verhalten einer solchen Schaltung, wie die des Inverters, einerseits am konkreten Objekt studieren und andererseits das Wesentliche daran, daß nämlich ein Null-Zustand an der Eingabe in einen Eins-Zustand an der Ausgabe verwandelt wird und ein Eins-Zustand an der Eingabe in einen Null-Zustand an der Ausgabe, in einer Tabelle ganz kurz und trocken notieren. *Bild 3* zeigt einfach an, was welchem Eingabewert an der Ausgabe durch die verwendete Schaltung zugeordnet wird. Der Inverter hatte an seiner Eingabe nur eine binäre Variable, den einen Schalter. Es gibt nun Schaltungen (die neben dem Inverter eine der Grundlagen der Computerei überhaupt bilden), die zwei oder mehrere Eingänge haben und die jeder

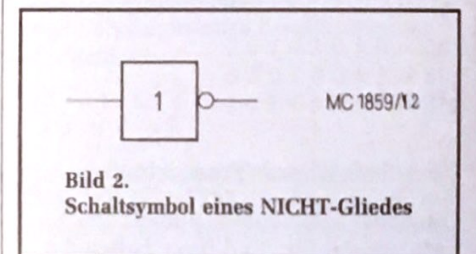


Bild 2. Schaltsymbol eines NICHT-Gliedes

Eingangskombination von Nullen und Einsen genau ein zugehöriges Ergebnis (0 oder 1) am Ausgang zuordnen. Zum Beispiel gibt es Schaltungen mit zwei Eingängen und einem Ausgang, wo genau dann eine 1 am Ausgang erscheint, wenn der eine Eingang UND der andere Eingang den Zustand 1 besitzen; in allen anderen Fällen erscheint eine Null am Ausgang. Eine solche Schaltung, sie kann in vielen Varianten aufgebaut werden, heißt ihrem Verhalten entsprechend UND-Schaltung, oder in der Fachsprache der Digitaltechniker UND-Glied. *Bild 3* zeigt sowohl die Zuordnungstabelle, die zum UND-Glied gehört, als auch seine abstrakten Schaltsymbole nach alter und neuer Norm. Neben diesem digitalen Schalt-Glied werden noch andere wichtige aufgeführt, deren Verhalten aus der zugehörigen Zuordnungstabelle abgelesen werden kann. Die Namen solcher Schaltglieder sind vom Verhalten abgeleitet. Solche Schalt-Glieder mit mehreren Eingängen heißen auch „Verknüpfungen", weil sie die Zustände der Eingänge hernehmen und zu einem Ausgangssignal verknüpfen.
Die wichtigsten drei Logikglieder sind das NICHT-, UND- sowie das ODER-Glied. Aus diesen Grundschaltungen lassen sich alle anderen Logikglieder zusammensetzen. So entsteht das NICHT-UND-Glied durch eine Reihenschaltung der Einzelglieder UND und NICHT. Mit diesen Logikgliedern und ihren Kombinationen werden im Computer Rechenvorgänge wie Addition und Subtraktion durchgeführt und Zahlen gespeichert.
Wir wollen nun zwei Einzelglieder näher betrachten, das UND-Glied und das Exklusiv-ODER-Glied. Beim UND-Glied besitzt der Ausgang nur dann den Zustand 1, wenn alle Eingänge den Zustand 1 besitzen (es können mehrere Eingänge vorhanden sein). Beim Exklusiv-ODER-Glied besitzt der Ausgang nur dann den Zustand 1, wenn nur einer der vorhandenen Eingänge den Zustand 1 hat.

Wieviel ist eins und eins?

Für binäre Rechenvorgänge, wobei meist aus zwei binär dargestellten Zahlen durch Verknüpfen eine neue entsteht, gibt es bestimmte Regeln, von denen wir uns kurz die für die binäre Addition anschauen. Zwei binäre Zahlen werden addiert, indem, mit dem niedrigsten Stellenwert beginnend, jedes Bit mit dem gleichwertigen Bit der anderen Zahl nach folgenden Regeln addiert wird:

0 + 0 ergibt 0
0 + 1 ergibt 1
1 + 0 ergibt 1
1 + 1 ergibt 0,

aber mit einem Übertrag 1.

Dieser Übertrag wird immer zum nächsthöheren Stellenwert addiert. Wir addieren die Zahlen 00111111 (63) und 00100110 (38) und beginnen mit den Ziffern ganz rechts. Das schaut dann so aus:

Neue Norm	Alte Norm	Beispiel	Wahrheitstafel	
E — 1 ○— A		7404	E | A 0 | 1 1 | 0	Nicht-Glied
E, E — & — A		7408	E E | A 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	Und-Glied
E, E — ≥1 — A		7432	E E | A 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	Oder-Glied
E, E — =1 — A	≠	7486	E E | A 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	Exclusiv-Oder-Glied
E, E — & — A		7400	E E | A 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	Nand-Glied-(Nicht-Und)
E, E — ≥1 ○— A		7402	E E | A 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	Nor-Glied-(Nicht-Oder)

Bild 3. Einige Logikglieder und ihre Wahrheitstafeln

Dezimalzahl	Binärzahl	
63	0 0 1 1 1 1 1 1	
+ 38	0 0 1 0 0 1 1 0	
11	1 1 1 1 1	Übertrag
101	0 1 1 0 0 1 0 1	

Es sei nun überlegt, wie man mit Logikgliedern eine solche Addition verwirklichen kann.
Vergleicht man die Rechenregeln mit den Wahrheitstafeln in Bild 3, so entsprechen diese der Verknüpfung eines Exklusiv-ODER-Gliedes, das die Summe zweier Bits bildet, mit einem UND-Glied, das den Übertrag ermittelt. Für jeden weiteren Stellenwert benötigt man aber den vorhergehenden Übertrag, der mit der Summe verknüpft werden muß.
Um zwei Bits zu addieren, brauchen wir also vier UND-, zwei NICHT- und drei ODER-Glieder. Für ein Byte brauchen wir dann acht solcher Logikgruppen.

Hexerei oder sedezimale Zahlen

Um nicht immer mit den langen Kolonnen aus Nullen und Einsen bei Binärzahlen arbeiten zu müssen, gibt es zur Vereinfachung der Zahlendarstellung das Sedezimal-System (Sechzehnersystem oder fälschlicherweise Hexadezimalsystem genannt). Dieses benutzt 16 Zeichen (die Ziffern 0–9 und die Buchstaben A–F) im Gegensatz zum Dezimalsystem, das 10 Ziffern (0–9) oder das Binärsystem, das nur zwei Ziffern (0 und 1) verwendet.
Die *Tabelle 2* des letzten Kapitels zeigt eine Gegenüberstellung der drei Stellenwertsysteme. Eine achtstellige Binärzahl ergibt eine nur zweistellige Sedezimalzahl.
Die Umwandlung einer Binärzahl in eine Sedezimalzahl ist einfach:
Man schreibt unter die Binärzahl von rechts beginnend für je vier Bit den Stellenwert, also die Potenzen der Zahl 2, und beginnt beim fünften Bit von vorne. Das sieht bei der Dezimalzahl 83 so aus:

0101 0011
8421 8421

Nun addiert man bei jeder Bit-Vierergruppe die Stellenwerte der Bits mit dem Wert 1, das ergibt die Zahlen 5 und 3. Die Sedezimalzahl lautet also 53.
Bei der Dezimalzahl 165 ergibt sich:

1010 0101
8421 8421
10 5

Die Zahl 10 muß in das sedezimale System umgewandelt werden, ergibt also nach Tabelle 2 den Buchstaben A. Die Sedezimalzahl lautet also A5.
Umkehrt läßt sich eine Sedezimalzahl leicht in eine Binärzahl umwandeln, da nur jede Stelle der Sedezimalzahl durch die dazugehörige Bitkombination ersetzt werden muß. Buchstaben muß man vorher in die Dezimalzahl verwandeln, und dann setzt man unter den in zwei Vierergruppen angeschriebenen Stellenwerten die Bits der Summanden jeder Zahl auf den Wert 1. Die Sedezimalzahl 4D ergibt dann:

4	D
4	13
8 4 2 1	8 4 2 1
0 1 0 0	1 1 0 1

Übrigens gibt es Taschenrechner, die Umwandlungen in verschiedene Zahlensysteme und auch Logikbefehle ausführen können.

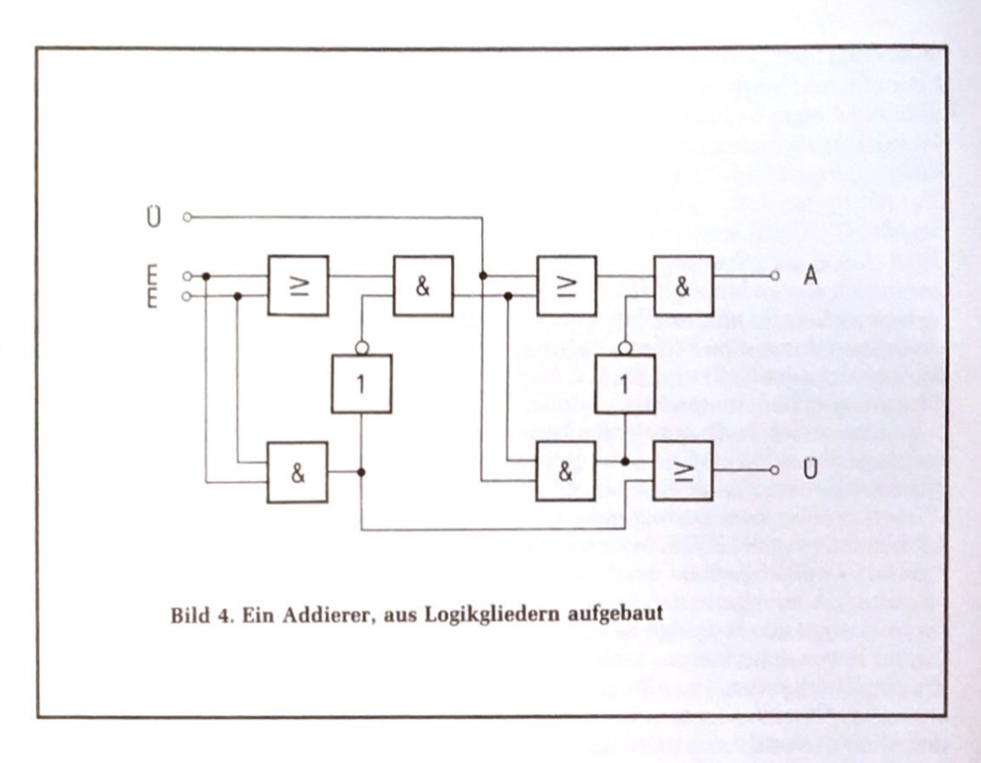
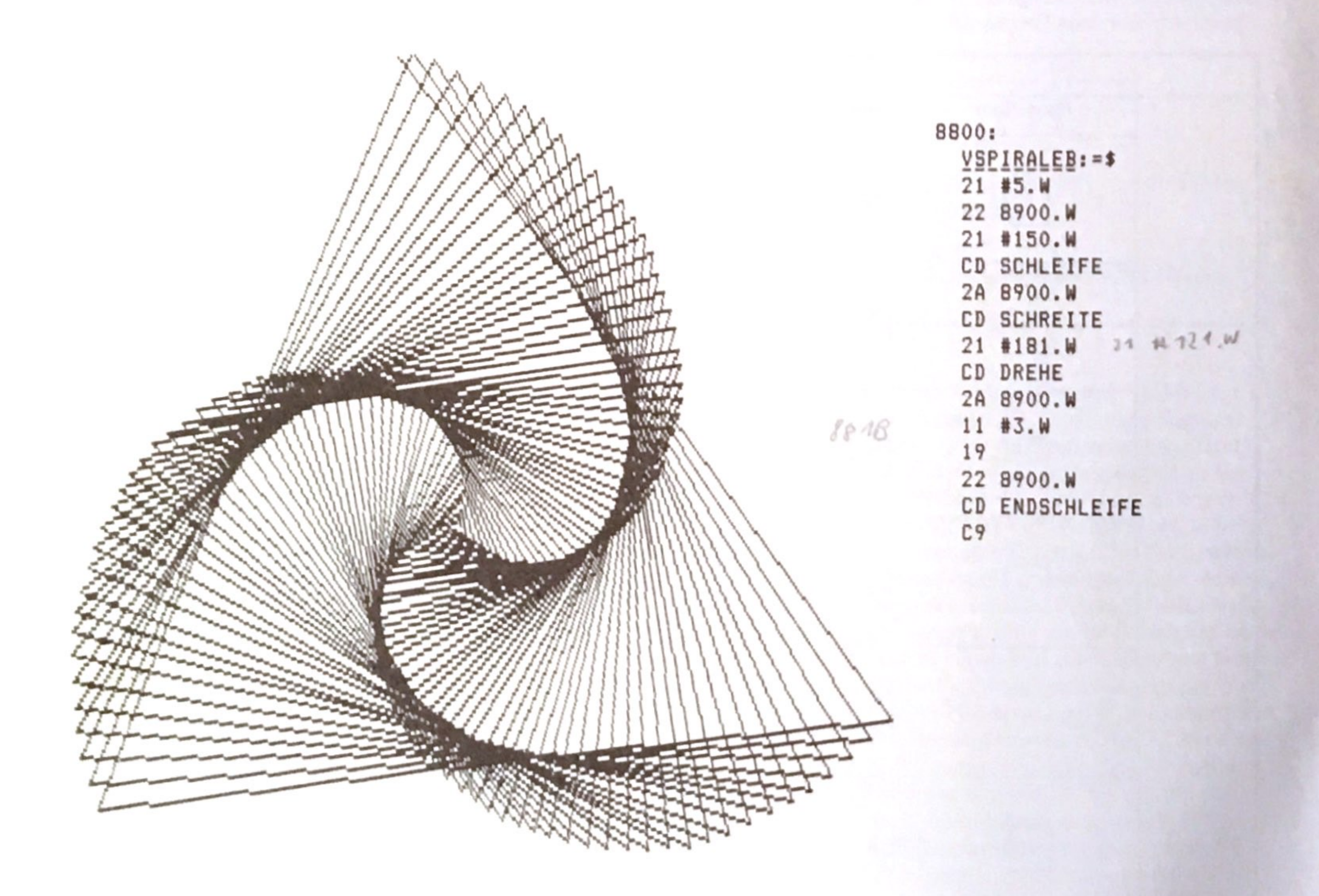


Bild 4. Ein Addierer, aus Logikgliedern aufgebaut



```
8800:
  VSPIRALEB:=$
  21 #5.W
  22 8900.W
  21 #150.W
  CD SCHLEIFE
  2A 8900.W
  CD SCHREITE
  21 #181.W
  CD DREHE
  2A 8900.W
  11 #3.W
  19
  22 8900.W
  CD ENDSCHLEIFE
  C9
```


Jürgen Plate

Vom Schaltplan zum Gerät

Mikroelektronik, Folge 3

Aller Anfang ist schwer – so heißt das Sprichwort. Ein Computer ist ein komplexes technisches Gerät, dessen Aufbau dem Anfänger einige Schwierigkeiten machen kann. Es ist ja nicht jeder unter Ihnen ein Elektronik-Profi. An einem einfachen Beispiel, einem Logikprüfstift, sollen Sie erste Übungen durchführen. Der Stift selbst ist einfach aufzubauen und das wichtigste Prüfinstrument beim Aufbau der anderen Schaltungen.

Jedes Gerät ist nach einem Schaltplan aufgebaut. In ihm ist festgelegt, welche Bauelemente verwendet werden und wie sie miteinander verbunden sind. Obwohl die Bauelemente bei gleicher Funktion recht unterschiedlich aussehen können, werden im Schaltplan genormte Schaltsymbole verwendet. So wird ein Widerstand durch ein Rechteck, ein Kondensator durch zwei parallele Striche dargestellt. Manche Bauteile besitzen eine bestimmte Orientierung, sie müssen in einer bestimmten Richtung eingebaut werden. Damit diese Orientierung erkennbar ist, werden sie im Schaltplan markiert: die Elektrolytkondensatoren besitzen eine mit + gekennzeichnete Elektrode, bei den Dioden ist die Katode (die Minusseite) durch die Richtung des Schaltsymboles markiert.
Doch zurück zum Schaltplan, der in *Bild* 1 wiedergegeben ist. Der Prüfstift besteht aus zwei integrierten Schaltungen, vier Widerständen und vier Leuchtdioden. Die erste integrierte Schaltung enthält vier Inverter, von denen wir nur drei I1, I2, I3 verwenden, die zweite ein Paar Flipflops, von denen nur eins (I1) verwendet wird. Der Schaltplan zeigt also nur die verwendeten Teile der ICs. Die Nummern an den Ein- und Ausgängen der Schaltungsteile der ICs entsprechen den Nummern der Beine des ICs.

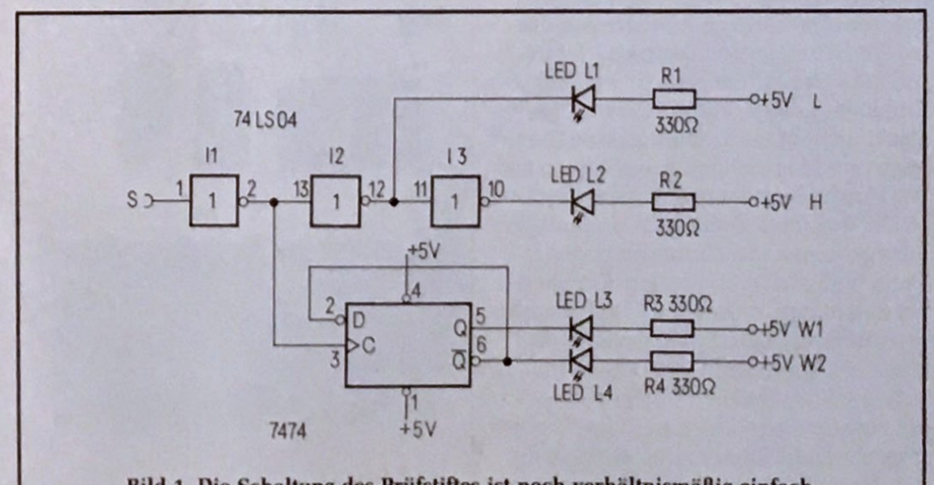


Bild 1. Die Schaltung des Prüfstiftes ist noch verhältnismäßig einfach

So funktioniert der Prüfstift

Das Prüfsignal gelingt von der Prüfspitze S an den Inverter I1, der den Rest der Schaltung vom Prüfobjekt trennt. Hinter I1 teilt sich das invertierte Eingangssignal. Im oberen Schaltungsteil wird das Signal mit I2 nochmals invertiert; hinter I2 steht also wieder das Originalsignal zur Verfügung. Von dort geht es zur Leuchtdiode L1 und zum Inverter I3. Ist das Eingangssignal bei S auf 0, leuchtet L1. L2 leuchtet genau dann, wenn das Eingangssignal bei S den Wert 1 besitzt. Die Widerstände R1 und R2 (ebenso R3 und R4) begrenzen den Strom durch die Leuchtdioden auf einen brauchbaren Wert.
Im Prüfstift wird auch ein sogenanntes Flipflop verwendet. Ein solches Schaltglied unterscheidet sich von den vorher schon besprochenen Logikgliedern insofern ganz elementar, als es nicht einfach irgendwelche Eingangssignale verknüpft, etwa wie ein UND aus zwei Signalen ein neues macht, sondern früher einmal vorgekommene Eingangssignale sich merkt und später zum Beispiel in Abhängigkeit davon auf neue Eingangssignale reagiert. Das Flipflop hier besitzt zwei Ausgänge, die so miteinander gekoppelt sind, daß der eine immer den inversen Zustand des anderen einnimmt. Das Flipflop kann nun, so ist es konstruiert, genau zwei Zustände annehmen, die nach außen dadurch sichtbar werden, daß der Ausgang Q einmal 0-Signal führt und einmal 1-Signal. Genau umgekehrt dazu liegen die Signale von Q̄. Das Flipflop, das hier verwendet wird, besitzt einen D-Eingang, der mit dem mit dem Pfeil gekennzeichneten Eingang zusammenspielt. Immer dann, wenn das Signal am mit dem Pfeil gekennzeichneten Eingang von 0 auf 1 wechselt, wird das in diesem Augenblick an D anliegende Signal übernommen und intern festgehalten. Q zeigt dann nach außen diesen neuen Zustand, Q̄ das Inverse dazu.
Die Rückführung des Signales von Q̄ an den D-Eingang ist nun ein raffinierter Kunstgriff. Während der kurzen Anstiegszeit des Signales am sogenannten „Triggereingang" wird zwar das Signal an D ins Innere des Schaltgliedes übernommen. Die Weiterleitung an den Ausgang Q und Q̄ geschieht aber mit einer wenn auch sehr kleinen Verzögerung, die gewährleistet, daß der von Q angeregte Wechsel des Zustandes, der ja auch Q wieder umsteuert, nicht mehr an D registriert wird. Erst bei einem neuen Anstieg der Signalflanke am Triggereingang spielt dieser neue Zustand eine

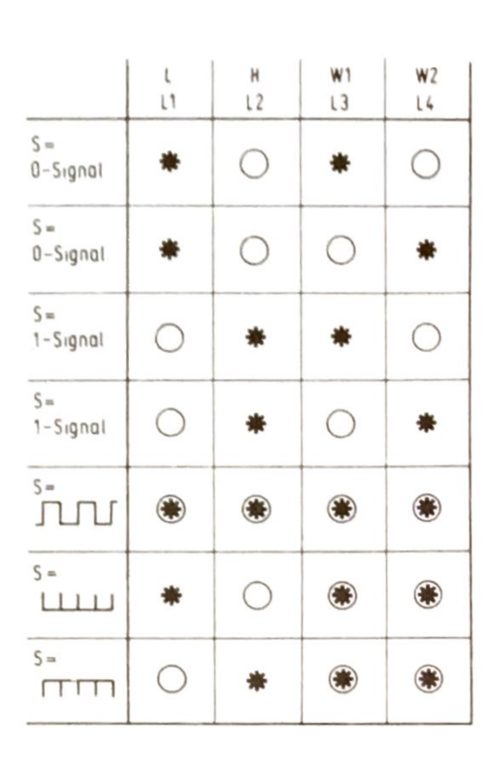
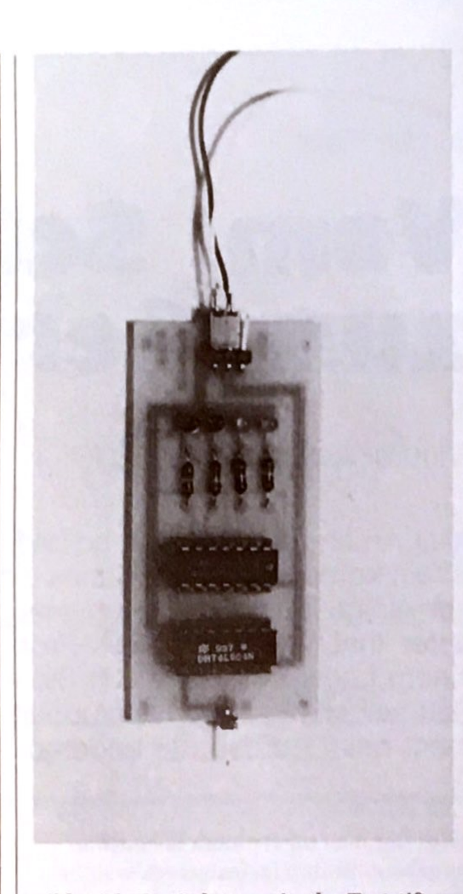


Bild 2. Das Schema, nach dem die Leuchtdioden am Prüfstift aufleuchten. Setzen Sie Rot für L und Grün für H ein, Gelb für die Ausgänge am Flipflop



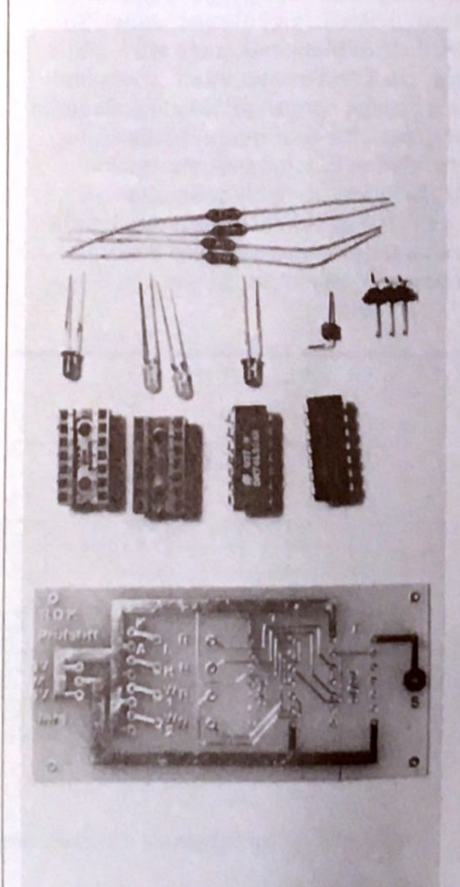


Bild 3. Die Bestückungsseite des Teststiftes mit den Bauelementen (oben). Die Lötseite (unten) mit den zurechtgelegten Bauelementen

Rolle. Das Fazit: Wenn am Prüfstifteingang S eine Folge von Rechteckimpulsen auftritt, also 01010101010..., dann ändert der Ausgang Q bei jedem Wechsel von 0 auf 1 an S seinen Zustand.
S: 01010101010...
Q: 01100110011...
Q: 10011001100
Jeder Wechsel des Eingangssignals von 0 auf 1 läßt das Flipflop kippen. So ergibt sich ein Signalwechsel am Ausgang des Flipflops; wenn zuvor L3 leuchtete, brennt nun L4 und umgekehrt. Während also der obere Teil statische Zustände anzeigt, werden mit dem Flipflop dynamische Vorgänge (Impulse) angezeigt. *Bild 2* zeigt alle möglichen Zustände des Prüfstifts.

Der Aufbau ist nicht schwer

Zunächst legen Sie die Teile des Bausatzes und das Werkzeug bereit. Meist brauchen Sie nicht mehr als einen Lötkolben mit gut verzinnter Spitze, Lötzinn, einen kleinen Seitenschneider und eine starke Pinzette. Lassen Sie sich Zeit mit dem Aufbau und arbeiten Sie sorgfältig – dann kann gar nichts schiefgehen. Die Platine hat eine Seite, auf der die Bauteile eingesteckt werden und eine Seite, auf der die Bauteileanschlüsse festgelötet werden. Die Lötseite (*Bild 3*) der Platine ist gekennzeichnet, meist mit dem Text „löts.". Nun werden zunächst die Fassungen für die integrierten Schaltungen eingelötet. Es gibt nun Fassungen, die eine kleine Kerbe an einer der beiden Schmalseiten besitzen. Damit soll gekennzeichnet sein, in welcher Einbaulage das IC auf die Fassung gesteckt werden muß, damit dessen Beine auch mit den richtigen Anschlüssen auf der Platine Kontakt bekommen. Es gibt leider aber auch Sockel, die undeutliche oder gar keine Markierungen tragen. Dann muß man sich vor dem Einbau des Sockels ansehen, wie das IC aufgesteckt werden muß (das ist meist auf der Bestückungsseite der Platine angezeichnet), weil der Sockel oft diese Kennzeichnung verdecken kann. Das IC selbst trägt ebenfalls immer eine Markierung, mit deren Hilfe Pin 1 aufgefunden werden kann. Leider sind diese Markierungen nicht bei allen Fabrikaten gleich.

Bild 4 zeigt schematisch ein paar Varianten. Man beginnt immer an der Seite zu zählen, auf der sich die Markierung befindet. Und zwar zählt man gegen den Uhrzeigersinn, wenn das IC von oben betrachtet wird. Zeigt darüber hinaus die Markierung an der Schmalseite auf den Betrachter, dann liegt Pin 1 rechts davon.

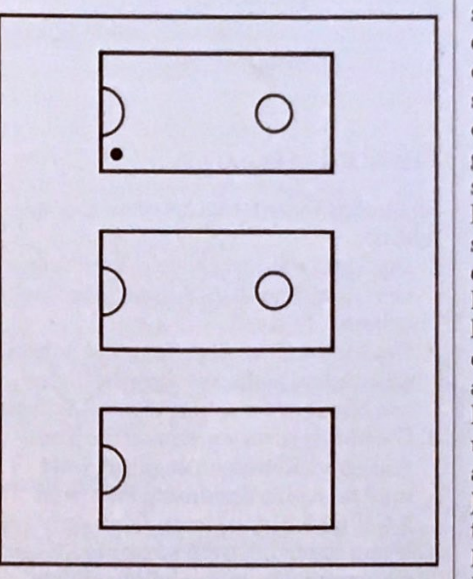


Bild 4. Verschiedene Gehäuseformen bei den ICs machen manchmal die Identifikation des Pin 1 schwer

Es wird gelötet

Zunächst löten Sie zwei diagonal gegenüberliegende Beinchen der Fassung an, damit sie nicht herausfallen kann. Danach werden dann alle anderen Beinchen angelötet. Die häufigsten Fehler beim Einlöten von IC-Fassungen sind einerseits Lötbrücken zwischen benachbarten Beinchen oder zwischen Beinchen und einer nahe gelegenen Kontaktierung auf der Platine, andererseits das Vergessen eines der Beinchen – achten Sie darauf. Das beste Prüfinstrument ist hier eine gute Lupe, mit der Sie auch kalte Lötstellen finden können. Anschließend werden die Steckkontakte eingelötet. Nun kommen die Widerstände und Kondensatoren dran. Beim Prüfstift gibt es keine Kondensatoren, trotzdem einige grundsätzliche Anmerkungen darüber: Es gibt ungepolte Kondensatoren, bei denen die Einbaurichtung keine Rolle spielt und Elektrolytkondensatoren, die mit der richtigen Polung eingebaut werden müssen. Die Elektrolytkondensatoren tragen dazu eine Markierung, die entweder auf den Plus- oder den Minuspol hinweist. Sie werden auch zwei Erscheinungsformen kennenlernen, die zylindrischen Becherelkos und die Tantalelkos, die wie Tropfen aussehen. Doch zurück zum Prüfstift. Jetzt werden die Widerstände in die Platine eingesteckt (wie herum ist egal), die Drähte auf der Lötseite etwas abgewinkelt und dann auf ca. 2 mm Länge abgeschnitten. Danach tritt wieder der Lötkolben in Aktion. Was nun noch fehlt, sind die Leuchtdioden. Diese müssen richtig herum eingebaut werden. Je nach Hersteller wird die Katode, die Minusseite, unterschiedlich markiert. Entweder, das Gehäuse ist ein wenig abgeplattet, das Anschlußbein ist länger als das andere oder es ist breiter als das andere. Sie finden die Katode, die an „Minus"

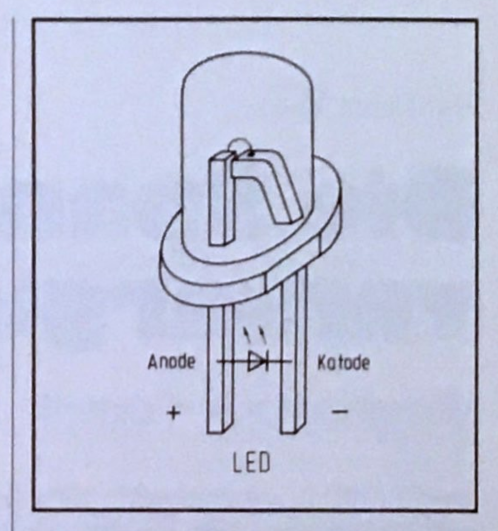
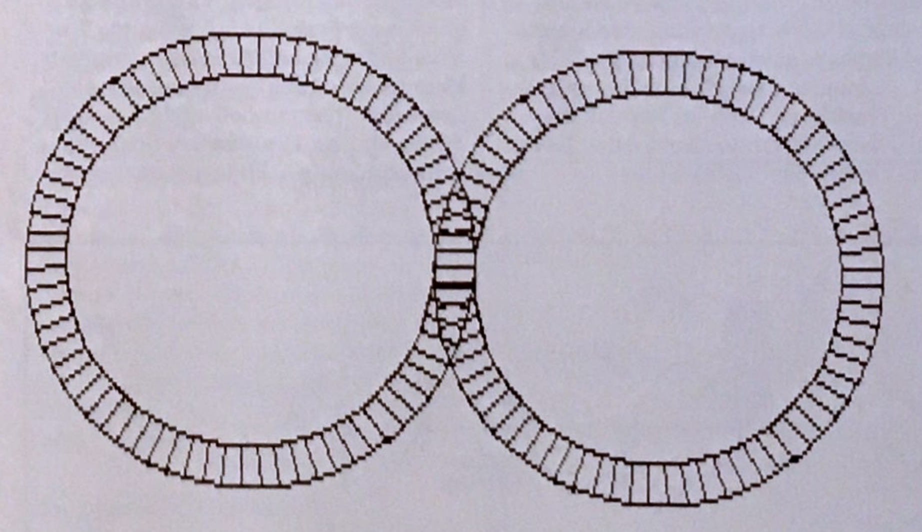


Bild 5. Die Katode einer Leuchtdiode ist immer die, die in ihrem Inneren gebogen ist oder größer als die andere ausgeführt ist

angeschlossen werden muß, sicher heraus, wenn Sie die Anschlüsse innerhalb der Leuchtdiode betrachten. Der „große, flächigaussehende", das ist die Katode (*Bild 5*).

Tips zur Fehlersuche

Fehler findet man am schnellsten durch systematisches Vorgehen. Beim Prüfstift gibt es beispielsweise zwei relativ unabhängige Einheiten. Ich will die Fehlermöglichkeiten nur in Stichpunkten aufzählen: keine Versorgungsspannung, IC oder Leuchtdiode verkehrt eingebaut, Lötbrücke zwischen zwei Kontakten, kalte Lötstelle, Lötstelle vergessen, Bauteil defekt.



```
8800:
  SCHIENEN:=$
  21 #2.W
  CD SCHLEIFE
  21 #72.W
  CD SCHLEIFE
  21 #8.W
  CD SCHREITE
  21 -#8.W
  CD SCHREITE
  21 -#90.W
  CD DREHE
  21 #20.W
  CD SCHREITE
  21 #90.W
  CD DREHE
  21 #11.W
  CD SCHREITE
  21 #90.W
  CD DREHE
  CD HEBE
  21 #20.W
  CD SCHREITE
  21 -#90.W
  CD DREHE
  21 -#3.W
  CD SCHREITE
  CD SENKE
  21 #5.W
  CD DREHE
  CD ENDSCHLEIFE
  21 #90.W
  CD DREHE
  21 -#20.W
  CD SCHREITE
  21 #90.W
  CD DREHE
  CD ENDSCHLEIFE
  C9
```

Rolf-Dieter Klein

Die Spannungsversorgung

Mikroelektronik, Folge 4

Die Spannungsversorgung für den NDR-Klein-Computer soll aufgebaut werden. Die Baugruppe heißt POW5V, gesprochen PO-Weh-5-Volt.

Es sei jetzt zuerst betrachtet, aus welchen Elementen ein Computer besteht: Da gibt es zunächst die „Eingabetastatur". Mit dieser Tastatur werden dem Computer die Wünsche der ihn nutzenden Personen mitgeteilt. Dann besitzt ein Computer fast immer einen Bildschirm als Ausgabeeinheit. Dort kann man wichtige Werte ablesen und zum Beispiel die Eingabe nochmals überprüfen.

Ferner gibt es fast immer einen Drucker, der das Ergebnis eines Laufes auf Papier festhält.

Der Computer besitzt im Zentrum eine „Datenverarbeitungseinheit", also den eigentlichen „Computer", und einen Datenspeicher. Der Datenspeicher enthält alle Informationen, die der Computer für seine Berechnungen benötigt.

All diese technischen Einheiten eines Computers werden heutzutage mit Strom betrieben, Sie benötigen also elektrische Energie.

Der Mikrocomputer, der hier im Heft geschildert wird und den jedermann preiswert aufbauen kann, besitzt im Endausbau ebenfalls alle genannten Einheiten. Man kann ihn Schritt für Schritt aufbauen. Das einfachste Element dabei ist die Spannungsversorgung, das Netzteil. Deshalb sei damit begonnen. Neben dem Lötwerkzeug wird auch ein Vielfachmeßinstrument mit Drehspulmeßwerk benötigt.

Energie für Computer

Mikrorechner sind sehr wählerisch, was die Energieversorgung angeht. Sie wollen eine oder mehrere Gleichspannungen haben, die bestimmte Werte genau einhalten müssen. So benötigt der NDR-Klein-Mikrorechner eine 5-V-Spannung, die sehr enge Toleranzen enthalten muß. Die Spannung darf nicht größer als 5,25 V sein, aber auch nicht kleiner als 4,75 V.

Wenn die Spannung nämlich zu groß wird, können Bauteile beschädigt werden. Ist sie zu niedrig, so arbeitet der Rechner nicht korrekt und liefert fehlerhafte Ergebnisse. Man kann also weder die Netzspannung von 220 V direkt für den Computer verwenden, noch kann man eine Taschenlampenbatterie als Energiequelle nutzen, da diese die geforderte Spannungstoleranz im Betrieb nicht einhält. Die Baugruppe POW5V (zusammen mit einem Netztransformator) löst das Energieversorgungsproblem.

Zunächst muß aus der lebensgefährlichen 220-V-Netzspannung eine harmlose Niederspannung von etwa 7,5 V bis 12 V gemacht werden. Es gibt eine Reihe von Transformatoren im Handel, die man dazu verwenden kann. Hier die Daten für den benötigten Trafo:

Eingangsspannung: 220 V
Ausgangsspannung: 7,5 V bis max. 12 V (am besten in Stufen einstellbar)
Leistung: ca. 30 Watt
oder Strom: ca. 3 Ampere
VDE-Zeichen.

Man kann ihn im Fachhandel oder bei den Baugruppen-Lieferanten besorgen. Am besten wäre eine Ausführungsform mit geschlossenem Gehäuse, bei der man die lebensgefährliche 220-V-Spannung nicht berühren kann.

Erstes Experiment

Folgendes Experiment ist dann ganz gefahrlos:

1. Der Trafo wird ans Netz angeschlossen (Achtung: VDE-Vorschriften beachten).
2. Das Meßgerät wird auf einen Wechselspannungs-Meßbereich gestellt, dessen Maximalausschlag über 12 V liegt.
3. Die Meßspitzen werden an die Trafo-Ausgangs-Klemmen angelegt. Jetzt muß man eine Spannung zwischen 7,5 V und 12 V ablesen können.
4. Wenn man ein Oszilloskop als Meßgerät verwendet, so erscheint auf dem Bildschirm eine sinusförmige Wechselspannung mit positivem und negativem Spannungsteil. Die Spannung zwischen der Nullinie und der Spitze der Wechselspannung ist höher als die auf dem Vielfachmeßgerät angezeigte Spannung.

Man bezeichnet die Spannung zwischen Nullinie und Spitze der Sinuskurve als Spitzenspannung, während ein Vielfachmeßgerät die sogenannte effektive Spannung anzeigt. Die Spitzenspannung steht nämlich nur ganz kurz während einer jeden Periode zur Verfügung. Das träge Meßwerk eines Zeigerinstruments kann nicht bis dahin ausschlagen, sondern registriert nur den effektiven Wert der Spannung. Die effektive Spannung kann man aus der Spitzenspannung ausrechnen. Dazu muß man den Wert der Spitzenspannung durch $\sqrt{2}$ (ungefähr 1,4142) dividieren.

$$U_{eff} = \frac{1}{\sqrt{2}} \cdot U_{spitze}.$$

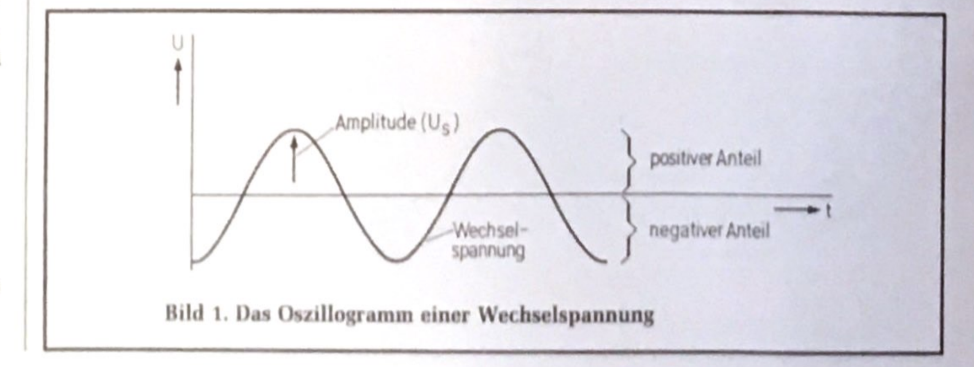


Bild 1. Das Oszillogramm einer Wechselspannung

Bild 1 zeigt den Verlauf einer Wechselspannung, wie man sie auf einem Oszilloskop sehen könnte. Jede Wechselspannung besitzt positive und negative Spannungsteile. Das ist für Gleichstrom-Geräte ungeeignet, da diese durch falsch gepolte Spannungen oft sogar beschädigt werden können. Ein Gleichrichter kann in solchen Fällen eingesetzt werden, um aus einer Wechselspannung eine Gleichspannung zu machen. Heute verwendet man dazu Brückengleichrichter, die aus vier Dioden bestehen.

Bild 2 zeigt, wie das Signal hinter dem Gleichrichter aussehen soll.

Bild 3 zeigt den Schaltplan der POW5V-Baugruppe,

Bild 4 den Bestückungsplan und *Bild 5* die fertige Baugruppe

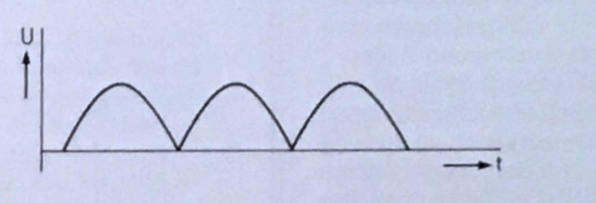


Bild 2. Ein Brückengleichrichter „klappt" die negativen Halbwellen der Wechselspannung nach oben um

Zum Aufbau

Der Gleichrichter wird als erstes auf die Leiterplatte POW5V gelötet. Dabei sollte man darauf achten, daß der Gleichrichter nicht verkehrt eingesteckt wird: Auf der Leiterplatte befindet sich eine Plus-Markierung, dort muß der mit „+" gekennzeichnete Gleichrichteranschluß eingesteckt werden. Der Transformatorausgang wird mit zwei Leitungen mit dem Wechselspannungseingang der Leiterplatte verbunden. Im Schaltbild ist noch eine Sicherung mit Si1 eingezeichnet. Diese Sicherung ist im allgemeinen im Trafo enthalten, wenn er VDE-mäßig aufgebaut ist (also mit Gehäuse und Anschlußschnur).

Die Sicherung ist daher nicht auf der Baugruppe eingeplant. Der Wechselspannungseingang der POW5V-Baugruppe ist mit einem Wellensymbol „~" gekennzeichnet.

Man kann die Leitungen von oben durch die Bohrungen der Leiterplatte stecken und unten auf der Lötseite verlöten. Man kann auch Lötstifte durch die Bohrungen stecken und die Zuleitungen an den Stiften festlöten. Wenn man 1,3-mm-Lötstifte verwenden will, muß man die 1-mm-Platinenbohrungen aufbohren. Es gibt aber auch 1-mm-Lötstifte im Handel, die man ohne Umstände einlöten kann.

Ein zweites Experiment

Mit dem Vielfachmeßgerät. Die Ausgangsspannung des Gleichrichters soll kontrolliert werden. Dazu wird das Meßgerät auf Gleichspannungsmessung eingestellt und ein Meßbereich über 20 V gewählt. Die Masseleitung des Meßgerätes, die mit „COM", „–" oder „0 V" am Gerät gekennzeichnet ist, wird mit dem Minusausgang des Gleichrichters verbunden. Der Plus-Eingang des Meßgerätes wird mit dem Plusausgang des Gleichrichters verbunden.

Der angezeigte Meßwert muß jetzt in etwa dem aus dem ersten Versuch entsprechen. Sollte ein Ergebnis ausbleiben, so kann man einen Widerstand parallel zum Ausgang des Gleichrichters schalten. Man nehme dazu zum Beispiel den Widerstand von 330 Ω, der im Bausatz vorhanden ist (aber später noch für einen anderen Zweck gebraucht wird).

Wenn sich jetzt kein befriedigender Ausschlag zeigt, dann sollte man alle Verbindungen nochmals genau überprüfen.

Wenn man ein Oszilloskop verwendet, muß man unbedingt diesen Widerstand (330 Ω, ¼ oder ⅛ W) parallelschalten. Ohne Widerstand fließt nämlich praktisch kein Strom durch den Gleichrichter. Das Meßergebnis wird deshalb verfälscht.

Mit dem Oszilloskop kann man erkennen, daß die Spannung am Gleichrichterausgang noch sehr wellig ist. Sie sinkt zwischendurch kurz auf Null Volt ab und steigt fast bis auf den Spitzenwert der Wechselspannung an.

Eine solche pulsierende Gleichspannung kann man mit einem Elektrolytkondensator, im Fachjargon auch als „Elko" bezeichnet, glätten. Beim Elko muß man beim Einbau darauf achten, daß er richtig gepolt wird, also der Plus-Anschluß des Elkos mit dem Plus-Ausgang des Gleichrichters verbunden wird. Auf der Leiterplatte ist das entsprechend markiert.

Wird ein Elko verkehrt herum eingebaut, so wird er zerstört. Im Schaltbild ist der Elko mit C1 bezeichnet.

Drittes Experiment

Mit einem Vielfachmeßinstrument: Die Ausgangsspannung hinter dem Elko liegt höher als bei der Messung ohne Elko (ca. um den Faktor 1.4), denn nun liegt eine fast glatte Spannung an. Der Elko wird bis zum Spitzenwert aufgeladen.

Messung mit dem Oszilloskop. Die Spannung am Elko ist praktisch eine glatte Linie. Auf dem Schirm sieht man jetzt normalerweise keine Welligkeit mehr. Wenn man aber eine Last (zum Beispiel eine Lampe) zum Elko parallelschaltet, so beginnt die vorher glatte Linie wieder Wellenform anzunehmen.

Der nächste Schritt ist der Einbau eines Spannungsreglers. Dieser hat die Aufgabe, aus der Gleichspannung von über 7,5 V eine exakte 5-V-Spannung zu machen.

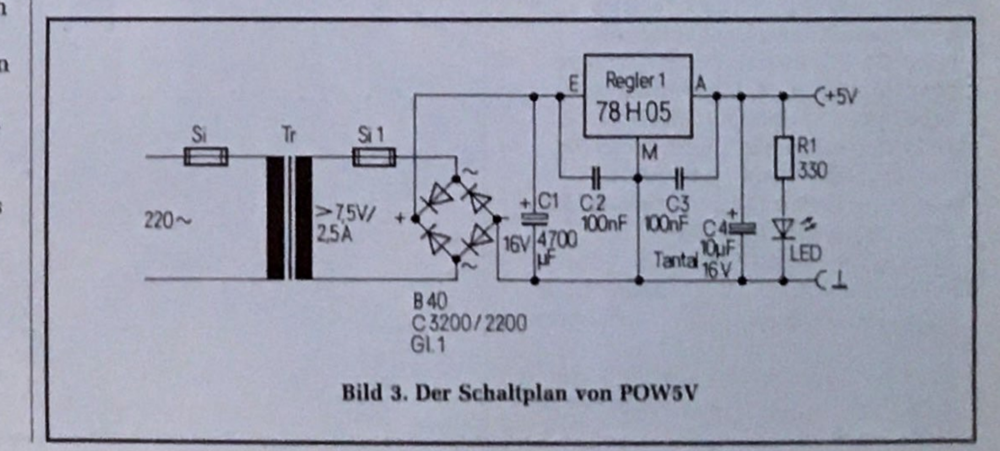


Bild 3. Der Schaltplan von POW5V

mikrocomputer schritt für schritt

So ein Spannungsregler, wie wir ihn verwenden, enthält in seinem Inneren eine Vielzahl von Transistoren und Widerständen, es ist ein integrierter Schaltkreis. Er prüft durch einen Spannungsvergleich in seinem Inneren, ob die Ausgangsspannung dem Soll entspricht, also exakt 5 V hat oder nicht. Ist die Spannung am Ausgang geringfügig abgesunken, so hebt er sie sofort wieder an und umgekehrt. Dazu benötigt er aber am Eingang eine Spannung, die größer sein muß als die von ihm geregelte Ausgangsspannung. Meist reichen 7,5 V effektive Eingangsspannung dazu gerade noch aus. Ist die Eingangsspannung niedriger als dieser Wert, so kann der Regler nicht mehr regeln, weil nicht mehr genug Spannungsreserve vorhanden ist, und am Ausgang sinkt die Spannung ab. Die Eingangsspannung darf also auch kurzzeitig nicht unter diesem Wert liegen, denn dann sinkt die Ausgangsspannung ebenfalls unter 5 V. Am Eingang des Spannungsreglers sollten also stets mehr als 7,5 V anliegen.

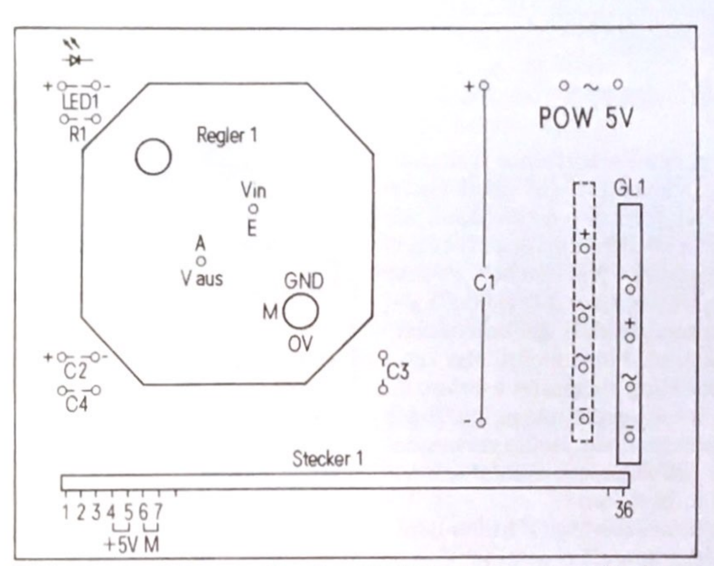


Bild 4. Der Bestückungsaufdruck zu POW5V. Da es zwei verschiedene Bauarten des Gleichrichters gibt, mit verschiedenen Anordnungen der Anschlußfahnen, sind zwei Einbaulagen gekennzeichnet. Der Gleichrichter sitzt richtig, wenn seine Markierungen der Anschlüsse mit den an den Platinenbohrungen für ihn übereinstimmen

Andererseits gibt es auch Schwierigkeiten, wenn zuviel Eingangsspannung am Regler anliegt. Der Spannungsregler muß die Spannungsdifferenz zwischen Eingang und Ausgang in seinem Inneren vernichten. Beispiel: Am Eingang liegen 9 V, am Ausgang 5 V. Die Differenz beträgt somit 4 V. Bei einem Strom von 1 A werden 1 A · 4 V in Wärme umgesetzt (U*I=P), also 4 W.
Die Energie, die der Spannungsregler in Wärme umsetzt, ist umso größer, je höher die Eingangsspannung ist und je mehr der Spannungsregler belastet wird, also je mehr Baugruppen angeschlossen sind. Dann wird der Regler einfach sehr sehr warm.
Eine im Regler eingebaute Temperatursicherung schaltet den Regler rechtzeitig ab, ehe er zu heiß wird. Jedoch liefert er dann auch keine 5 V mehr, der angeschlossene Computer bleibt stehen. Es ist also gar nicht so einfach mit dem Regler. Jedoch nicht verzagen, Ausprobieren zeigt, daß alles gut funktioniert. Der Regler wird auf einem Kühlkörper montiert, der die Wärme besser an die Umwelt abführen soll. Auf der POW5V-Baugruppe werden Kühlkörper und Regler mit Schrauben befestigt.
Achtung: Die Anschlußbeinchen des Reglers dürfen den Kühlkörper nicht berühren, ggf. sollte man Isolierhülsen verwenden. Die Anschlußbeine sind am Regler asymmetrisch angebracht. Der Regler paßt also nur in einer Lage bequem auf die Platine.

Zum sicheren Betrieb des Reglers werden noch drei kleinere Kondensatoren benötigt. Im Schaltbild sind sie mit C2, C3 und C4 bezeichnet. Die Kondensatoren C2 und C3 sind 100-nF-Kondensatoren, deren Polung keine Rolle spielt, und C4 ist ein kleiner Elektrolytkondensator oder ein sogenannter Tantalkondensator von 10 µF, der die Ausgangsspannung von Störungen befreien soll. Bei seinem Einbau ist wieder die Plusmarkierung zu beachten.

Ein viertes Experiment

Es wird am Vielfachinstrument der 5-V-Gleichspannungsbereich (oder der nächst verfügbare höhere Bereich) eingestellt. Man sollte jetzt eine Ausgangsspannung von sehr genau 5 V messen. Der Wert darf minimal bei 4,75 V und maximal bei 5,25 V liegen, größere Abweichungen sind nicht zugelassen.
Als Krönung des Ganzen gibt es im Bausatz noch eine Leuchtdiode und einen Widerstand. Die Leuchtdiode wird an die Stelle LED 1 eingelötet. Dabei muß man auf die Polung der Leuchtdiode achten. Das längere Bein ist der +-Anschluß.
Danach wird noch der Widerstand R1 eingelötet, der als Schutz für die Leuchtdiode dient. R1 hat den Wert 330 Ω. Die Farbringe zeigen dann orange-orange-braun-gold. Gold ist der sogenannte Toleranzring, der anzeigt, daß der Widerstand höchstens um 5 % vom angegebenen Wert abweicht. Man kann auch 10%-Widerstände mit silbernem vierten Ring verwenden.

Wenn man die Leuchtdiode verkehrt herum eingelötet hat, so leuchtet sie nicht. Sie wird davon nicht zerstört, man sollte sie aber nicht zu lange in diesem Zustand lassen. Eine Steckleiste mit abgewinkelten Pfostensteckern „im 2,54-mm-Raster" wird am Schluß auf der Bestückungsseite mit den abgewinkelten Stiften eingesteckt und dann eingelötet. Wenn man keine passenden Stiftleisten bekommt, kann man auch längere einfach passend abschneiden oder aus kurzen eine lange konstruieren. Bei der POW5V sind nur 4 Stifte wirklich belegt, die anderen dienen der mechanischen Stabilität, wenn man die POW5V-Baugruppe später in ein Buchsenfeld steckt. Die Stiftleiste dient später als Verbindung zu den restlichen Baugruppen des NDR-Klein-Computers. Den vollständigen Aufbau zeigt *Bild 5*.

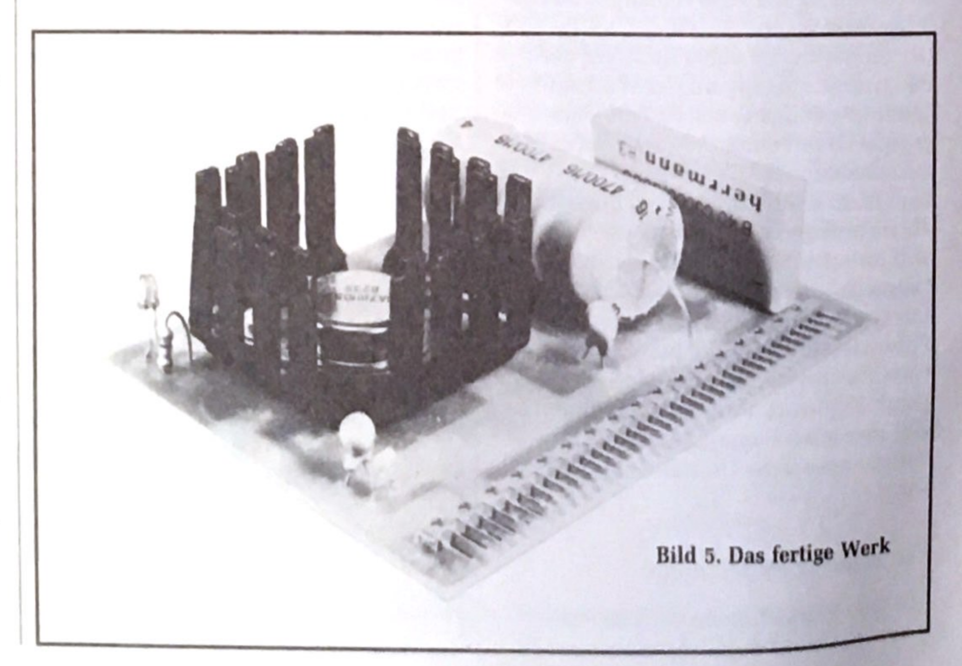


Bild 5. Das fertige Werk

Zusatzaufgaben als Anregung für Lehrer

1. Meßreihe. Die Ausgangsspannung in Abhängigkeit von der Ausgangslast, z. B. Lastströme von 100 mA, 1 A, 2 A, 3 A, die man durch unterschiedliche Widerstände oder mit einem Regelwiderstand erreichen kann.
Der Spannungsregler ist kurzschlußfest, daher kann nichts passieren. Die Widerstände können heiß werden, wenn sie zu wenig Leistung vertragen (5 V mit 3 A Last entspricht 15 W Leistung).
2. Meßreihe. Die Ausgangsspannung in Abhängigkeit von der Eingangsspannung. Am Wechselspannungseingang wird 2 V, 4 V, 5 V, 6 V, 7 V, 8 V, 9 V eingestellt und am Ausgang einmal ohne zusätzliche Last und einmal mit 1-A-Last gemessen.



Rolf-Dieter Klein

Die erste Aufbaustufe: Startlogik und Taktgenerator

Mikroelektronik, Folge 5

Es wurde gesagt, daß ein Computer aus verschiedenen Elementen besteht: Aus der Eingabeeinheit, einer Ausgabeeinheit und aus einer Verarbeitungseinheit mit Speicher. Jedes dieser Elemente ist selbst wieder aus Elementen aufgebaut. Jetzt sei die Verarbeitungseinheit des Computers genauer unter die Lupe genommen.

Bild 1 zeigt schematisch die Elemente, aus welchen unsere Verarbeitungseinheit besteht. Da gibt es:

1. Eine Startlogik. Sie hat die Aufgabe, den Rechner zum Beispiel nach dem Einschalten zu starten. Und zwar muß der Rechner da mit einer ganz bestimmten Einstellung zu arbeiten anfangen.
2. Einen Taktgeber. Er liefert den Arbeitstakt für den Rechner. Ein Rechner muß nämlich seine Befehle in einem genau festgesetzten Rhythmus abarbeiten. Dazu benötigt er einen stabilen Takt. Je höher dieser Takt ist, desto schneller ist der Rechner bei seiner Arbeit. Ist der Takt zu hoch, so werden allerdings seine Schaltkreise überfordert. Jeder Rechner besitzt daher eine genau festgesetzte Takt-Frequenz, die für ihn optimal ist.
3. Die Zentraleinheit. Das ist der eigentliche Rechnerbaustein. Er wird oft CPU genannt, das ist die Abkürzung für Central Processing Unit, also auf deutsch: Zentraleinheit. In diesem Baustein laufen alle Operationen ab, die das Wesen eines Rechners ausmachen.

Als Zentraleinheit wird hier der Baustein mit der technischen Bezeichnung Z80-A verwendet. Der Buchstabe A gibt an, daß die CPU mit 4 MHz arbeiten kann. Neben den A-Bausteinen gibt es auch B-Bausteine (6 MHz), L-Bausteine (7,5 MHz) und H-Bausteine (8 MHz). Die Arbeitsfrequenz bestimmt, wie schnell Befehle ausgeführt werden können und damit, wie schnell Berechnungen durchgeführt werden können. Im folgenden

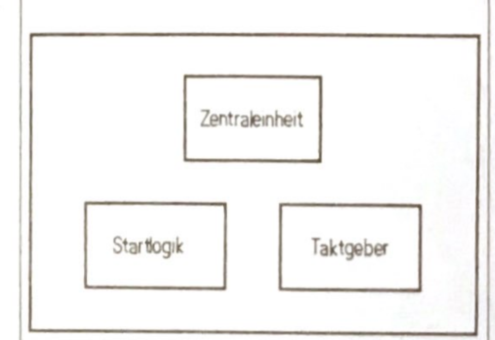


Bild 1. Im Herzen des Computers arbeitet die Zentraleinheit (CPU). Sie wird von einem Taktgeber angetrieben und von einer Startlogik richtig gestartet

wird der Buchstabe A stets weggelassen, da die weiteren Angaben unabhängig vom jeweiligen Typ sind.
Der Z80-Baustein, das ist also ein Vertreter der berühmten Mikroprozessoren. Es ist ein sehr bewährter Typ, der in der Industrie sehr häufig eingesetzt wird. Eine Platine mit dieser CPU sei jetzt Schritt für Schritt aufgebaut.

Erster Schritt: Wie funktioniert die Startlogik?

Zum einen soll der Rechner richtig starten, sobald die Spannung eingeschaltet worden ist. Zum anderen muß man ihn per Taste neu starten können, wenn man den Rechner dazu bringen will, daß er eine einmal begonnene Rechnung abbrechen soll und von vorn neu starten soll. Dazu wird ein mechanischer Taster verwendet, dessen Schaltsignal ausgewertet wird. Der Z80, unsere Zentraleinheit, besitzt einen separaten Eingang für solch ein „Rücksetz-Signal", der allerdings nicht direkt mit einem mechanischen Taster verbunden werden kann. Das hat folgenden Grund: Wenn ein mechanischer Taster betätigt wird, so berühren sich dessen Kontaktflächen mehrere Male kurz hintereinander. Man sagt, der Taster prellt.
Dieses Prellen entsteht, weil die Kontaktzungen wie Federn wirken und beim Aufeinanderprallen zu schwingen anfangen. Die Zeitdauer eines solchen Prellvorgangs liegt in der Größenordnung von Millisekunden (1 ms = $1/1000$ s), wobei noch die Bauart des Tasters eine Rolle spielt. Der Z80 würde bei einem solchen Signal an seinem Rücksetzeingang sehr durcheinander geraten. Daher muß eine Zusatzschaltung, die Startlogik, den Taster ergänzen. Diese Startlogik soll, wie schon gesagt, dafür sorgen, daß auch nach dem Spannungseinschalten erst einmal ein Startsignal an die Zentraleinheit gegeben wird.
Als erstes wollen wir also den Taster entprellen, also dafür sorgen, daß beim Betätigen nur ein Signalwechsel erfolgt. *Bild 2* zeigt eine Schaltung, bei der ein Anschluß des Tasters mit dem 0-V-Anschluß (Masse) verbunden ist. Der andere Anschluß ist an einen Widerstand (R1) und an einen Kondensator (C1) geführt. Der Widerstand ist mit seinem anderen Ende an +5 V angeschlossen und der Kondensator an 0 V (Masse).
Im Ruhezustand, also wenn die Taste nicht betätigt ist, kann sich der Kondensator C1 über den Widerstand R1 auf +5 V aufladen. Wird der Taster nun betätigt, so wird der Kondensator sehr schnell entladen, und die Ausgangs-

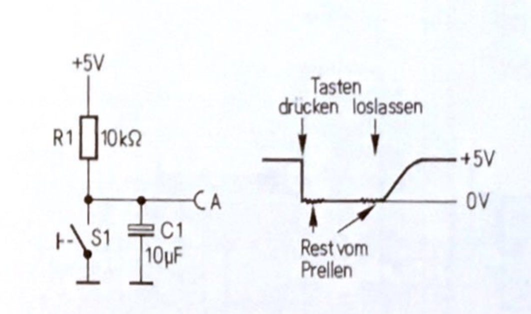


Bild 2. Ein Taster, der mit einem RC-Glied entprellt wird. Ohne das RC-Glied würden im Moment der Betätigung des Tasters kurze Spannungssprünge zwischen 0 und 5 V auftreten. Ebenfalls beim Loslassen des Tasters

spannung an Punkt A geht auf 0 V zurück. Wenn nun die Kontakt-Zungen des Tasters hin- und herfedern, so bedeutet das: die Taste wird praktisch ein paarmal kurz losgelassen. Der Kondensator verhindert, daß die Spannung am Kontakt in diesen kurzen Zeiten auf 5 V ansteigt; er wird über R1 so langsam aufgeladen, daß die Spannung praktisch ständig 0 V bleibt.
Auch beim Loslassen schließt der Taster noch ein paarmal nach dem ersten Öffnen. Wieder kann sich der Kondensator in dieser Zeit nicht auf +5 V aufladen. Erst wenn das letzte Prellen aufgehört hat, wird er sich langsam auf +5 V aufladen. Der Taster ist also mit einem R-C-Glied entprellt worden.

Nun hat aber diese einfache Schaltung in bezug auf den Z80 doch einen Haken. Das Signal steigt sehr langsam auf +5 V an. Der Z80 zum Beispiel weiß mit solchen Signalen nicht viel anzufangen, seine digitalen Eingänge verlangen nach ihrer Konstruktion ein sehr schnell ansteigendes Signal, da sonst die interne Schaltung nicht richtig arbeitet. Wir müssen also unser Signal noch weiter aufbereiten.

Der nächste Schritt

Das langsam ansteigende Signal muß in ein schnell ansteigendes Signal umgewandelt werden. Dazu wird ein sogenannter Schmitt-Trigger verwendet. Dieser Baustein ist genau dafür konstruiert, sich langsam ändernde Signale so umzuformen, daß „steile" Umschalt-Flanken entstehen.

Im Bausatz gibt es dazu den integrierten Baustein mit der Typenbezeichnung 74121. Dieser Baustein enthält einen solchen Schmitt-Trigger. Er enthält aber auch noch ein anderes interessantes Element, nämlich ein sogenanntes Monoflop. Bitte lassen Sie sich jetzt von der Vielfalt an neuen Begriffen nicht verwirren: Die Ingenieure haben lange gebraucht, um sich das alles auszudenken. Wichtig ist, was das Monoflop tut. Es macht aus einem Eingangssignal einen immer gleich breiten Ausgangspuls. Genauer gesagt, wenn beim Monoflop im Inneren des 74121-Bausteins eine Signalflanke eintrifft, die von 0 auf 5 V ansteigt, dann antwortet es an seinem Ausgang mit einem Impuls, dessen Länge von außen genau einstellbar ist.
Bild 3 zeigt die Schaltung und die Signalformen.

Das Signal A ist das Eingangssignal, das in den integrierten Schaltkreis 74121 eingegeben wird. In dem Schaltkreis sind ein Monoflop und ein Schmitt-Trigger in Reihe geschaltet. Am Punkt A1, der allerdings von außen nicht zugänglich ist, da er in der integrierten Schaltung verborgen liegt, ist das Signal nach der Bearbeitung durch den Schmitt-Trigger eingezeichnet. Am Punkt A2 tritt das Signal nach dem Monoflop auf. Es liegt normalerweise auf 5 V und geht, wenn der Taster losgelassen wurde, eine kurze, genau festgelegt Zeit auf 0 V. Die Zeitdauer, die im Bild mit t bezeichnet ist, wird durch einen Kondensator (C2) bestimmt. Dieses Signal kann nun direkt an die Zentraleinheit geführt werden. Von der Form her würde zum Betrieb der Zentraleinheit das Signal A1 ausreichen. Jedoch würde der Z80 nicht arbeiten, solange der Taster gedrückt ist. Das würde später beim weiteren Ausbau stören. Daher wird das Monoflop verwendet, um den Z80 nur für kurze Zeit außer Gefecht zu setzen.

Bild 4 zeigt den gesamten Schaltplan der SBC2-Baugruppe. *Bild 5* zeigt den Bestückungsplan und *Bild 6* die Stückliste. Erschrecken Sie nicht, es wird alles Schritt für Schritt aufgebaut und erklärt.

Jetzt wird gelötet

Erster Schritt: Wir löten den Sockel für IC1 ein. Achtung, so ein Sockel besitzt meist eine kleine Marke an einer der Schmalseiten. Diese Marke soll in Richtung auf die Steckerleiste zeigen. Den Sockel kann man nur sinnvoll einlöten, wenn man darauf achtet, daß er nicht auf

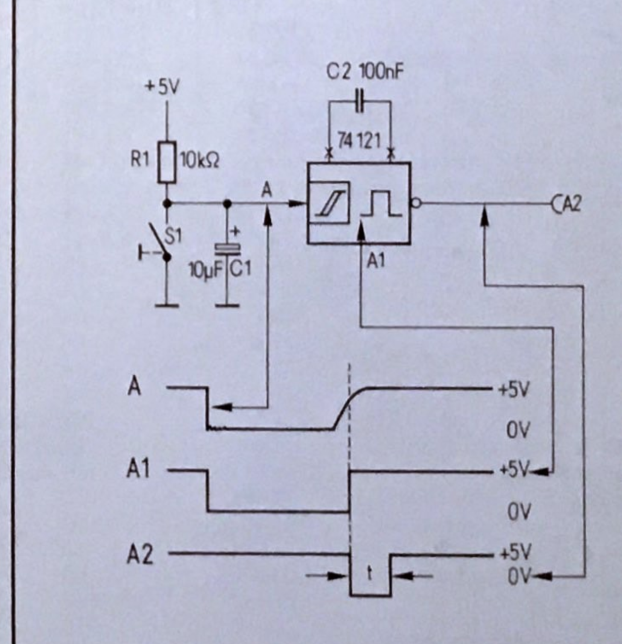


Bild 3. Schaltung und Signal-Zeit-Plan der Startlogik. An Punkt A liegt das Signal aus Bild 2 an. An Punkt A1 im Inneren des ICs könnte man das vom Schmitt-Trigger in ein sauberes „Rechteck" verwandelte Signal beobachten. An Punkt A2 liefert der im IC enthaltene „Monoflop" (deutsch etwa: Einmal-Impulsgeber) sein Ausgangssignal. Er wird dazu von der ansteigenden Flanke von Signal A1 angestoßen (getriggert)

der Lötseite, sondern auf der Bestückungsseite der Leiterplatte eingesteckt wird, so daß die Beine, die Pins, auf der Lötseite der Platine herausragen.
Diese Seite, auf der ausschließlich gelötet wird, ist mit „Löts." am Platinenrand beschriftet.
Zweiter Schritt: Der Widerstand R1 wird eingelötet. Er besitzt einen Wert von 10 kΩ (Farbringe: braun-schwarz-orange-gold (oder silber).
Dritter Schritt: Der Kondensator C1 wird eingelötet. Es handelt sich um einen Tantalkondensator (Perlenform), manchmal auch um einen Elektrolytkondensator, mit einem Wert von 10 µF. Dabei ist entweder der Pluspol oder der Minuspol des Kondensators gekennzeichnet, und man muß beim Einbau unbedingt darauf achten, daß die Polung stimmt: Plus zur Plusmarkierung oder Minus zur Minusmarkierung!
Vierter Schritt: Der Kondensator C2 wird eingelötet. Dabei handelt es sich um einen 100-nF-Kondensator, bei dem die Polung keine Rolle spielt.
Fünfter Schritt: Ein Taster wird über zwei isolierte Litzen mit der Leiterplatte verbunden.
Sechster Schritt: Die 36polige Stiftleiste wird auf der Bestückungsseite eingesteckt und eingelötet.

Siebter Schritt: Nun wird die Spannungsversorgung POW5V mit der SBC2 gekoppelt. Man kann dazu die noch nicht geschilderte Bus-Baugruppe verwenden oder in diesem Fall auch einfach zwei Leitungen anlöten. Dabei darf man keinen Fall vorn auf die Stifte der Stiftleiste löten, da man diese sonst später nicht mehr stecken kann. Man sollte die Stromversorgungszuleitungen auf der Lötseite der Leiterplatte anlöten. Dazu wird der +5-V-Ausgang von POW5V mit dem +5-V-Eingang der SBC2-Leiterplatte verbunden. Ebenso der 0-V-Ausgang mit dem 0-V-Eingang (Masse) der SBC2-Leiterplatte. (Das sind jeweils zwei Lötaugen mit der Bezeichnung +5 V und M.)
Achter Schritt: Nun wird die Versorgungsspannung eingeschaltet. Mit einem Vielfachinstrument (oder wahlweise Oszilloskop) mißt man die ankommende Spannung. Dabei wird der Minuspol des Instruments mit Pin 7 des IC-Sockels verbunden und der Pluspol mit Pin 14. Die Zählweise ist dabei so, daß, von oben gesehen, wenn die IC-Fassung mit der Markierung auf den Betrachter zeigt, der Pin rechts neben der Markierung Pin 1 ist. Der Pin mit der höchsten Nummer liegt dann links der Markierung. Es muß eine Spannung von +5 V ankommen. Ist das nicht der Fall, so liegt entweder ein Kurzschluß vor, dann leuchtet auch die LED auf der POW5V nicht mehr; oder man hat versehentlich die Anschlüsse verpolt (dann wird eine negative Spannung angezeigt); oder es liegt irgendwo eine Unterbrechung vor, dann leuchtet die LED, aber keine Spannung wird angezeigt; oder man mißt am falschen Punkt des IC-Sockels, dann schaue man sich einmal die IC-Belegung und deren Numerierung an.
Neunter Schritt: Nun kann man noch an Pin (Anschlußbein) 5 des Sockels IC1 messen. Dort muß das entprellte Tastersignal anliegen. Wenn man die Taste drückt, so sinkt die Spannung schnell auf 0 V, wenn man die Taste losläßt, so steigt sie etwas langsamer wieder auf +5 V an. Mit dem Vielfachinstrument kann man dies natürlich nicht so genau verfolgen wie mit einem Oszilloskop.

Zehnter Schritt: Spannung wieder ausschalten. Nun wird der integrierte Schaltkreis eingesetzt. Dabei muß man

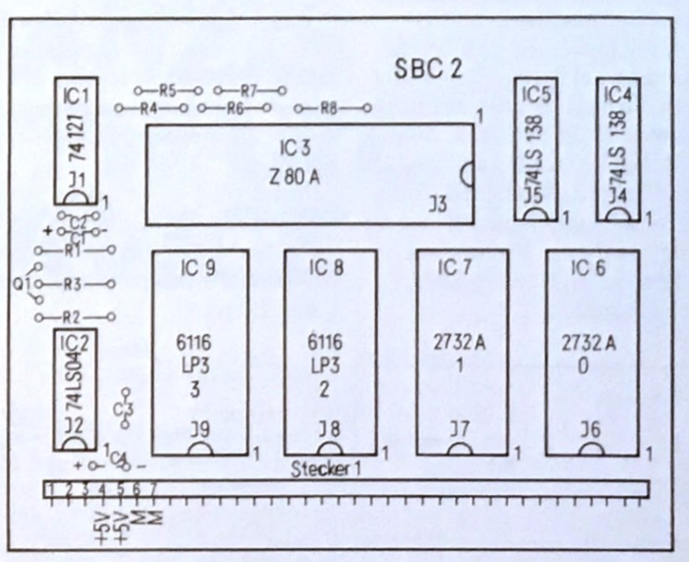


Bild 5. So liegen die Bauelemente auf der Platine. Pin 1 der ICs ist jeweils angemerkt. Von da an zählt man nach hinten, und von dort auf der anderen Seite wieder nach vorn

J1=IC1	74 121	
J2=IC2	74 LS 04	keinen 74 04 verwenden!
J3=IC3	Z 80 A	
J4=IC4	74 LS 138	
J5=IC5	74 LS 138	
J6=IC6	(2732 A, Grundprogramm, Basic oder Gosi)	
J7=IC7	(2732 A, Grundprogramm, Basic oder Gosi)	
J8=IC8	6116 LP3	
J9=IC9	6116 LP3	
R1	10 kΩ	
R2	1 kΩ	
R3	1 kΩ	
R4	1 kΩ	
R5	1 kΩ	
R6	1 kΩ	
R7	1 kΩ	
R8	300 kΩ	
C1	10 µF	
C2	100 NF	
C3	10 NF	
C4	10 µF	
Q1	4,000 MHz	
Stecker1	36polig	
S1	Taster	

Hinweis:
Im Buch ist ein 100-nF-Kondensator als C3 angegeben. Wir verwenden an dieser Stelle einen 10-nF-Kondensator. Es funktionieren beide Versionen.

Bild 6. Die Stückliste zur SBC2-Platine

Bild 4. Die Gesamtschaltung der Single-Board-Computer-Baugruppe. Hier im Kapitel werden nur die Teile ganz links aufgebaut

wieder auf die Orientierung achten. Auf dem integrierten Schaltkreis ist eine Marke auf der Schmalseite angebracht. Diese Marke muß bei IC1 zur Steckerleiste hinzeigen, also mit der des Sockels übereinstimmen (Bild 5). Wenn man sich nicht sicher ist, sollte man mal in Kapitel 3 nachsehen. Die Beschriftung des ICs gibt keine Anhaltspunkte auf die Orientierung. Sind zwei Marken auf dem IC vorhanden, so ist die *größere* Marke entscheidend.

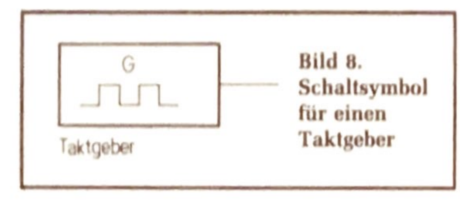


Bild 8. Schaltsymbol für einen Taktgeber

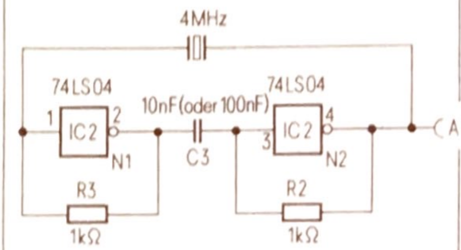
Elfter Schritt: Die Spannung wird wieder angelegt. An Pin 1 des IC1 muß der entprellte kurze Puls ankommen, wenn man die Starttaste drückt. Diesen Puls kann man mit einem Prüfstift messen. Dazu wird dieser ebenfalls mit der Spannungsversorgung verbunden (zuvor abschalten) und dann der Prüfeingang an Pin 1 des ICs 74121 angelegt. Wenn man die Taste drückt, so muß jeweils genau ein Wechsel der LEDs von W1 auf W2 oder umgekehrt erfolgen. Dann arbeitet die Schaltung. Der Puls ist so schmal, daß man ihn mit dem Skop nur sehr schwer erkennen kann. Daher ist hier der Prüfstift das beste Testmittel.

Jetzt kommt der Taktgeber dran

Ein Taktgeber ist nötig, weil ein Computer in Schritten arbeitet, beinahe wie ein Uhrwerk. Der Taktgeber teilt der Zentraleinheit mit, wann und in welcher Geschwindigkeit die einzelnen Verarbeitungsschritte auszuführen sind.

Kernstück des Taktgebers ist ein Schwingquarz. Dieser Schwingquarz liefert, betrieben mit der richtigen Beschaltung, eine sehr konstante Schwingung. Die Schaltung rund um den Quarz besteht ebenfalls aus einem integrierten Bauteil.

Bild 9 zeigt den inneren Aufbau der Schaltung. Es zeigt zwei Nicht-Glieder (N1 und N2), einen Kondensator (C3), zwei Widerstände (R2 und R3) und einen Quarz Q.



9. Die hier verwendete Schaltung zur Takterzeugung

Nehmen wir einmal an, am Eingang von N1 (Pin 1) liegt ein 0-Signal vor, also 0 V. Dann invertiert das Nicht-Glied diesen Wert und liefert am Ausgang eine 1, also +5 V (bei ICs kann dieser Wert auch kleiner sein, minimal 2,5 V). Diese Spannung gelangt nun über den Widerstand R3 wieder zurück an den Eingang des Nicht-Gliedes. Diesmal aber als 1-Signal. Das wird nun wieder invertiert, und am Ausgang liegt ein 0-Signal an, das wieder an den Eingang gelangt und so weiter und so fort. Durch die Schwingungen des Nicht-Glieds wird nun auch der Quarz angeregt, seinerseits zu schwingen. Der Quarz kann aber (genau das ist sein Zweck) nur auf einer bevorzugten Frequenz schwingen. Daher kontrolliert der Quarz die Schwingungen des Nicht-Gliedes. Das Ergebnis ist ein stabiler Takt.

Das Nicht-Glied N2 arbeitet synchron mit N1 und gibt das Taktsignal weiter. Quarzfrequenzen gibt man meist in MHz an. Unser Quarz soll eine Frequenz von 4 MHz (vier Millionen Schwingungen pro Sekunde) liefern. Man könnte auch einen anderen Quarz verwenden, zum Beispiel 2 MHz. Dann arbeitet die CPU (Zentraleinheit) aber langsamer.

Und wieder Löten

Wir wollen die Schaltung nun aufbauen. Im Gesamt-Schaltbild *Bild 4* findet man die Taktgeber-Schaltung wieder. Zum Aufbau:

1. Sockel des IC2 74LS04 einlöten und dabei auf die Marken achten.
2. Die beiden 1-kΩ-Widerstände R2 und R3 einlöten (Farbringe: braun-schwarz-rot).
3. Den Kondensator C3 einlöten. Als Werte sind 10 nF oder 100 nF verwendbar. Der Kondensator besitzt keine Polung. Daher spielt es keine Rolle, wie man ihn einlötet.
4. 4-MHz-Quarz einlöten. Der Quarz wird liegend eingebaut (Bild 7).
5. Das IC 74LS04 wird in die Fassung IC2 eingesteckt. Hier unbedingt wieder auf die Orientierung (Markierung IC und Markierung des Sockels) achten, denn ein falsch eingestecktes IC wird mit Sicherheit zerstört.
6. Die Spannung einschalten.
7. Mit dem Prüfstift wird an IC2, Pin 8, gemessen. Es müssen alle vier Leuchtdioden des Prüfstifts (H, L, W1 und W2) aufleuchten. Dann ist die Schaltung ok.
8. Man kann die Messung mit einem Oszilloskop auch genauer durchführen. An Pin 8 muß eine Frequenz von 4 MHz anliegen.
   Die Signalform ist dabei nicht so sehr entscheidend. Bei 4 MHz wird man kein exaktes Rechtecksignal mehr erhalten, da die Frequenz sehr hoch ist.
9. Fehlersuche. Wenn sich nichts tut, kann man, wenn man in einer Arbeitsgruppe arbeitet, das IC probeweise einmal mit dem des Nachbarn tauschen.
   Wird das IC sehr heiß, so wurde es wahrscheinlich falsch herum eingesteckt. Man kann dann versuchen, das IC nochmals richtig einzusetzen, meist ist es jedoch zerstört. Daher gut aufpassen beim Einsetzen von integrierten Schaltungen.
   Wenn die Schaltung bei korrekt eingesetztem IC nicht arbeitet, so kontrolliere man einmal die Lötseite der Leiterplatte. Meist liegt dann ein Lötfehler vor. Man kann zum Beispiel zwei Anschlußbeinchen versehentlich miteinander verlöten.

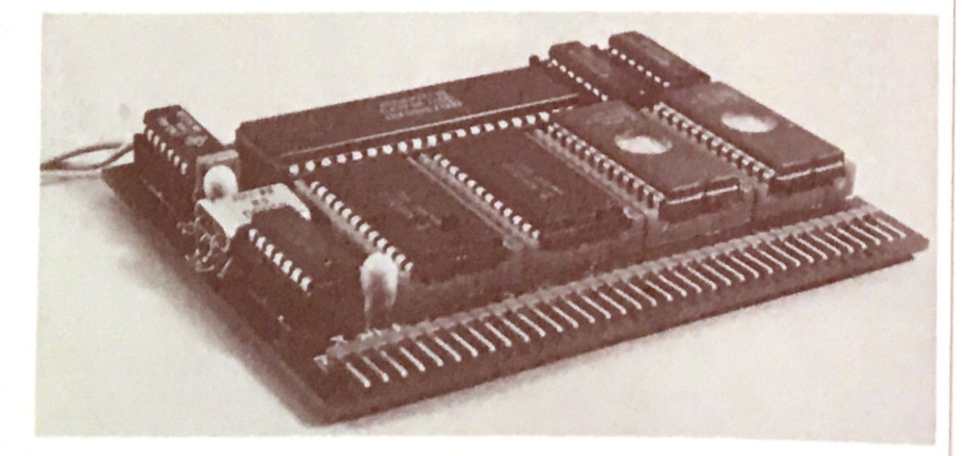


Bild 7. So sieht die fertig aufgebaute Platine aus

Rolf-Dieter Klein

Die Zentraleinheit wird eingesetzt

Mikroelektronik, Folge 6

Inzwischen werden Sie schon ein bißchen Routine gewonnen haben. Sie können mit einem „Zisch" ein IC-Sockelbein einlöten. Sie können fachmännisch von Elkos sprechen und Sie wissen, wo Pin 1 eines ICs sitzt. Es werden noch viele Dinge auf Sie einstürmen, die neu sind. Aber Sie werden feststellen, daß das Verstehen der Mikroelektronik sehr viel Freude macht.

Die SBC2-Baugruppe ist im vorherigen Abschnitt nicht fertiggestellt worden. Also weiter im Text:

Man löte alle restlichen Fassungen nach Bestückungsplan ein. Dabei auf die Markierungen achten!

Alle restlichen Widerstände (vier 1-kΩ-Widerstände mit dem Farbcode braun-schwarz-rot und 1 Widerstand 330 Ω mit dem Farbcode orange-orange-braun) jetzt einlöten.

Nun wird der Prozessor Z80-A feierlich in die 40polige Fassung eingesetzt. Bitte unbedingt auf die Orientierung achten. Und wenn es nicht gleich klappt, dann biegt man die Beine zurecht, indem man das IC in beide Hände nimmt, zwischen Daumen und Zeigefinger, und auf einer ebenen Unterlage mit sanftem Nachdruck alle 20 Beine einer Seite gleichzeitig biegt.

Spannung einschalten. Mit dem Prüfstift an Pin 6 der CPU messen. *Bild 1* zeigt die Z80-CPU mit ihren Anschlüssen. Bei Pin 6 steht die Bezeichnung Takt. Dort muß also der 4-MHz-Takt des Taktgebers erscheinen. Beim Prüfstift leuchten alle vier LEDs (H, L, W1 und W2) auf, wenn alles korrekt funktioniert. Wenn man mit einem Oszilloskop arbeitet, so kann man auch nochmals die Frequenz überprüfen. Sie beträgt 4 MHz. Die Periodendauer beträgt also 250 ns (Nanosekunden).

Jetzt den Prüfstift an Pin 26 anschließen. Wenn man den RESET-Taster betätigt, so muß entweder die LED W1 ausgehen

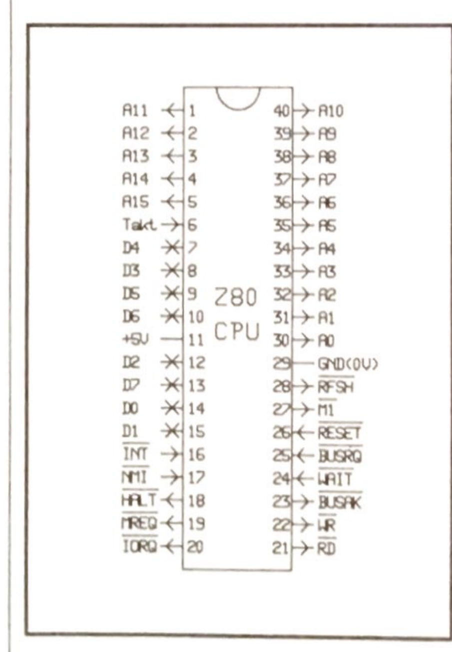


Bild 1. Das sind die Anschlüsse der CPU, also der Zentraleinheit. Beachten Sie, daß alle Steuersignalbezeichnungen einen Querstrich tragen. Das bedeutet, daß sie bei Low-Pegel aktiv sind

und W2 an, oder W2 aus und W1 an. Da bis auf kurze Impulse ein 1-Signal an diesem Anschluß liegt, leuchtet die LED H. Der Eingang ist der RESET-Eingang der Zentraleinheit. Er ist direkt mit dem Ausgang der Start- und Rücksetzlogik verbunden.

Nun an Pin 11 messen. Dort muß die LED H leuchten, denn dort muß die +5-V-Versorgung anliegen. An Pin 29 muß Masse anliegen und somit leuchtet die LED L, wenn man hier mit dem Prüfstift mißt.

Nun kann man den Prüfstift an die Anschlüsse 19, 20, 21 und 22 legen. Welcher Signal-Wert dort anliegt, hängt von den Umständen ab (*Bild 2*). Da auf der Platine noch wesentliche Bauelemente fehlen, sind die Reaktionen an den Ausgängen der CPU undefiniert.

Wenn man Bild 1 nochmals ansieht, so sind alle Anschlüsse der CPU mit Pfeilen versehen. Bei Ausgängen weisen sie von der CPU weg, bei Eingängen zur CPU hin. Dann gibt es aber auch noch acht Leitungen, die durch Doppelpfeile gekennzeichnet sind. Auf diesen Leitungen können Informationen sowohl in die CPU hinein als auch aus der CPU heraus transportiert werden.

Was ist ein Befehl?

Jetzt kommt etwas Wichtiges: Wenn der Taktgeber läuft und gerade am RESET-Eingang ein korrekter Impuls anlag, dann dauert es nur wenige Takte, bis die CPU an den Leitungen D0 bis D8 abtasten möchte, welche Pegel dort anliegen. Sie ist so gebaut, daß sie damit erfahren will, was sie als erstes tun soll. Sie will einen Befehl bekommen. Betrachten Sie nochmals Bild 1. Die CPU erhält also alle Befehle über die sogenannten Datenleitungen. Diese Datenleitungen, die alle Doppelpfeile tragen, sind im Bild mit D0, D1, D2, D3, D4, D5, D6 und D7 bezeichnet. Wenn man alle diese Leitungen auf 0 V legt, so heißt das für die CPU: „tue nichts". Für erste Experimente sei ein „Nichts-tu-Stecker" aufgebaut. Man kann dazu einen Stecker in die 24polige Fassung stecken, in die später unser Speicher kommt. Zuvor muß der Nichts-tu-Stecker aber noch geeignet verdrahtet werden. *Bild 3* zeigt, wie.

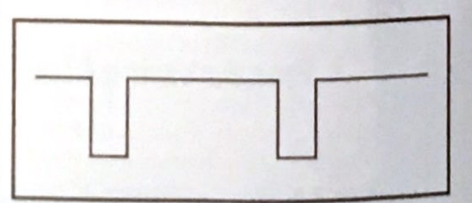


Bild 2. In bestimmten Abständen tauchen auf der RD-Leitung Impulse auf, die anzeigen, daß der Prozessor lesen möchte

Man achte dabei auf das T-Symbol in der Zeichnung, das die Orientierung angibt. Links neben dem T-Symbol liegt Pin 1. Folglich werden, da man Pins immer gegen den Uhrzeigersinn zählt, die Pins 9, 10, 11, 12, 13, 14, 15, 16 und 17 miteinander verbunden. An Pin 12 liegt später Masse.

Der Stecker wird dann (Achtung: Einbaulage beachten!) in die Fassung 0 der SBC2-Karte gesteckt. Man könnte für dieses Experiment auch jede andere Fassung verwenden, da die Datenleitungen an alle Fassungen geführt sind. Diese Datenleitungen sind über Leiterbahnen auf der Leiterplatte mit den entsprechenden Anschlüssen der CPU verbunden. D0 ist zum Beispiel an Pin 9 aller 24poligen Fassungen auf der SBC2-Baugruppe, D7 an Pin 17 zu finden. Man kann dies mit einem Durchgangsmesser prüfen (Vielfachinstrument auf Widerstandsmessung).

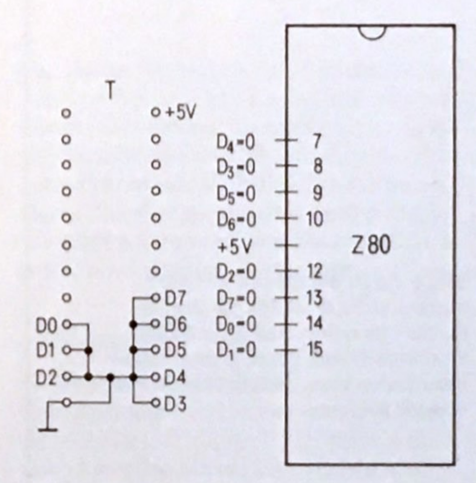


Bild 3. Mit diesem Schaltungsvorschlag wird dem Z80 an seinen acht Datenleitungen konstant Low-Pegel, also 0, angeboten. Der Prozessor interpretiert das als Nichts-tu-Befehl, der in der Fachsprache NOP heißt

Immer wieder experimentieren

Wenn man die Spannung eingeschaltet und den RESET-Taster betätigt hat, sollte man folgende Messungen durchführen:

1. Mit dem Prüfstift an Pin 19 der CPU. Alle vier LEDs des Prüfstifts müssen leuchten.
2. Wir messen an Pin 21 der CPU. Auch hier müssen alle vier LEDs des Prüfstifts leuchten.
3. Wir messen an Pin 20 der CPU. Die LED H muß leuchten und dann noch W1 oder W2.

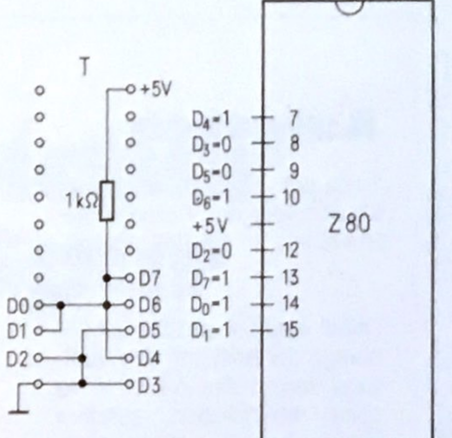


Bild 4. Beim OUT-Befehl werden 11010011 auf den Datenleitungen erwartet

4. An Pin 22 darf nur die LED H leuchten und eine der LEDs W1 oder W2.

Mit dem Oszilloskop kann man die Signale genau ansehen. An Pin 19 und an Pin 21 erscheinen kurze Pulse, wie in Bild 2 sichtbar.

Was hat es mit diesen Leitungen auf sich? In Bild 1, neben Pin 19, steht die Beschriftung MREQ. Das bedeutet Memory-Request, zu deutsch Speicher-Anforderung.
Wenn die CPU auf dieser Leitung Pulse aussendet, so will sie etwas vom Speicher. Was sie genau will, sagt sie jedoch nicht allein mit diesem Signal.

Pin 21 ist mit RD bezeichnet. Das bedeutet Read, also Lesen. Wenn die CPU hier Pulse ausgibt, so will sie etwas lesen.

Jetzt wird auch die Bedeutung der Doppelpfeile an D0 bis D7 klarer. Wenn Pulse auf RD anliegen, so können in diesen Augenblicken Daten oder Befehle von außen über die Leitungen D0 bis D7 ins Innere der CPU gelangen.
Bei Pin 22 steht die Beschriftung WR. Das bedeutet Write, also Schreiben. Wenn hier Pulse anliegen, so will die CPU in diesen Augenblicken Daten über D0 bis D7 nach außen übertragen. Die Datenleitungen D0 bis D7 können also sowohl Daten von außen nach innen übertragen als auch umgekehrt. Fachleute sprechen in solch einem Fall von bidirektionalen Datenleitungen.
Nun noch zu Pin 20. Dort steht IORQ. Das bedeutet Input/Output-Request. Gemeint ist, daß die CPU mit der Außenwelt in Verbindung treten will, diesmal aber nicht mit dem Speicher, sondern mit anderen Schaltungsteilen, die aus dem Computer heraus führen oder hinein. Man spricht von Peripherie.

Was die CPU tut, wenn sie nichts tut

Solange der Nichts-tu-Stecker im Sockel 0 steckt, wird man nur auf der RD-Leitung und auf der MREQ-Leitung Pulse feststellen. Klar, denn die CPU wollte weder etwas schreiben noch wollte sie etwas mit der Außenwelt zu tun haben, denn sie hatte ja gerade aufgegeben bekommen, „nichts" zu tun.
Immer, wenn sie diesen Befehl bekommt, der aus lauter Lows (0 V) auf den Datenleitungen besteht, wartet sie eine genau definierte Anzahl von Takten und tut nichts weiter dabei. Dann fragt sie erneut nach einem Befehl und trifft in unserem Fall wieder auf den Nichts-tu-Befehl, der ja fest verdrahtet ist. Jedesmal wenn RD aktiv ist erfährt sie also, daß sie immer noch nichts tun soll.

Von Leitungsbezeichnungen und weiteren Befehlen

Mancher mag sich vielleicht über die Querstriche über den Bezeichnungen MREQ und anderen Leitungen wundern. Der Querstrich sagt, daß die Leitung im Ruhezustand ein 1-Signal führt. Bei IORQ und WR kann man auch ein 1-Signal mit dem Prüfstift (oder Oszilloskop) messen, solange der Prozessor nichts tut. Der Querstrich ist ein Hinweis des CPU-Herstellers, der die Orientierung erleichtern soll. Man sagt auch, daß solche Signalleitungen „low-aktiv" sind.

Wir können nun noch zwei weitere Versuche durchführen. *Bild 4* und *Bild 5* zeigen zwei weitere Verdrahtungen von Steckern.
Der Stecker nach Bild 4 ist so verdrahtet, daß er den OUT-(Output oder Ausgabe) Befehl an die CPU liefert. Mit diesem Befehl wird die CPU aufgefordert, Daten an die Außenwelt zu liefern. Es sollte jetzt schon immer klarer werden, daß die Zentraleinheit intern so aufgebaut ist, daß sie zu bestimmten Taktzeiten die Bitmuster, die ihr auf den acht Datenleitungen angeboten werden, übernimmt und intern als Befehl auswertet. Wenn man IORQ, WR, MREQ und RD mißt, so werden jetzt an allen vier Leitungen Pulse erkennbar sein (beim Prüfstift werden alle Leuchtdioden leuchten).
Der Stecker nach Bild 5 ist ein IN-Stecker. Er veranlaßt die CPU Daten von der Außenwelt aufzunehmen. Bei einer Messung werden die Leitungen IORQ, RD und MREQ Pulse zeigen und WR wird ein 1-Signal führen.
Hier ein Hinweis: Ein 1-Signal besteht beim Z80 meist nicht aus einem Signal

von genau 5 V. Der genaue Wert ist sogar von IC zu IC verschieden. Er liegt zwischen +2,5 V und 5 V. Ein 0-Signal liefert dagegen eine Spannung zwischen 0 V und 0,7 V.
Ein Prüfstift erkennt diese Pegel richtig. Mit dem Oszilloskop kann man die Sache genauer betrachten. Dabei zeigen sich oft noch merkwürdige Dinge auf dem Bildschirm: Den eigentlichen Pulsen sind oftmals kleinere Impulse überlagert. Für die Funktion sind sie ohne Bedeutung, solange die Gesamtspannung in den Spannungsbereichen 0 V bis 0,7 V und 2,5 V bis 5 V bleibt.

Wir fassen nochmals zusammen:
Die CPU braucht Befehle zum Arbeiten. Diese Befehle gelangen über die Datenleitungen D0 bis D7 ins Innere der CPU. Die CPU besitzt spezielle Leitungen, über die sie den angeschlossenen Bausteinen mitteilt, ob sie Daten haben will oder welche ausgeben will, ob die CPU mit dem Speicher zu tun haben will oder mit der Außenwelt.

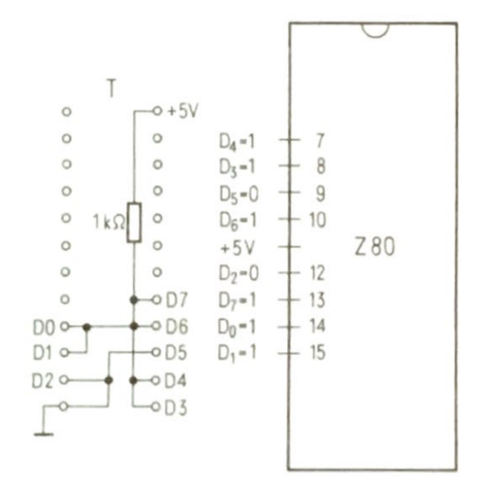


Bild 5. Der IN-Befehl besteht aus dem Bitmuster 11011011

Aufgaben

1. Es soll ein Stecker gebaut werden, der das Datenmuster 01110111 an die CPU liefert.

Dabei wird D7 und D3 mit 0 V belegt, der Rest mit +5 V. Achtung, man sollte die +5 V nie direkt anschließen, sondern immer über einen Widerstand, da sonst Kurzschlüsse entstehen, wenn die CPU Daten ausgibt. Ein Kurzschluß nach 0 V ist hingegen unschädlich.
Der Stecker muß an RD, MREQ und WR Pulse liefern, IORQ muß auf 1 bleiben.

2. Was tut der Stecker also?
3. Man sollte sich die anderen Leitungen der CPU ansehen. Wie ist die Reaktion bei verschiedenen Steckern?

Interessant ist beim NOP-Befehl das Spiel auf den Leitungen A0 bis A15. Während die A-Leitungen mit niedrigen Nummern ständig blitzschnell die Pegel ändern, wird dieses Spiel nach oben hin immer langsamer. Dort kann man den Wechsel mit bloßem Auge verfolgen.



Bild 6. So ist der Stecker für NOP aufgebaut. NOP ist die Abkürzung für No Operation. Solch ein Befehl ist sinnvoll, weil damit genau festgelegte Zeiten ohne Aktion überbrückt werden können

HALLO HALLO

```
8800:
  TEXT:=$          STARTEHIER:=$     CD SCHREITE
  #160.W           21 TEXT1          21 -#90.W
  #130.W           CD WRITE          CD DREHE
  33.B             21 #60.W          21 #1.W
  0.B              CD SCHLEIFE       CD SCHREITE
  "HALLO"          21 #120.W         21 -#90.W
  0.B              CD SCHREITE       CD DREHE
  TEXT1:=$         21 #90.W          CD ENDSCHLEIFE
  #300.W           CD DREHE          CD WAIT
  #130.W           21 #1.W           3E 01.B
  33.B             CD SCHREITE       D3 70.B
  0.B              21 #90.W          21 TEXT
  "HALLO"          CD DREHE          CD WRITE
  0.B              21 #120.W         C9
```


Rolf-Dieter Klein

Dem Speicher auf der Spur

Mikroelektronik, Folge 7

Im Grunde war das, was im vorherigen Kapitel geschildert wurde, etwas gemein, weil der Prozessor Z80 mit den Steckern um seine liebste Beschäftigung gebracht wurde: Er unterhält sich am liebsten mit Speicherbausteinen; einerseits möchte er deren Inhalt erfahren und andererseits je nach Verlauf der Rechnungen auch etwas dorthin schreiben.

Die leeren Sockel auf der Platine SBC2 werden in Kürze mit höchstintegrierten ICs gefüllt werden. Es werden dort Speicherbausteine Platz finden, die dem Prozessor nicht nur NOP-Befehle mitteilen. Bevor das aber getan wird, muß einiges über Speicher gesagt werden, damit klar wird, was diese Bausteine tun.

Ordnung muß sein

Es gibt Leute, bei welchen es oben auf dem Speicher des Hauses wie „bei Hempels im Garten" aussieht. Dementsprechend werden diese Leute nichts wiederfinden. Andererseits gibt es Ordnungsfanatiker, die auch das geringste Stück etikettiert haben und darüber Buch führen, wo sie es aufheben. Mikrocomputer neigen ebenfalls zur Pedanterie. Allerdings etikettieren sie nicht das einzelne Stück, das sie aufheben wollen, sondern die Stellen, in welchen man etwas aufheben und wiederfinden kann. Jedem Speicherplatz ist eine Nummer zugeteilt.
Im vorigen Kapitel war als Übungsaufgabe die Überprüfung des Spieles auf den A-Leitungen gestellt worden. Sechzehn davon gibt es beim Z80. Dabei sollten Sie festgestellt haben, daß die Leitung A15 etwa im Sekundenrhythmus ihren Pegel wechselt, während A14 dies doppelt so geschwind tut und A13 dies viermal so schnell. Auch bei den Leitungen, bei welchen man das mit dem Prüfstift und dem bloßen Auge nicht mehr so einfach feststellen kann, ist es so, daß die mit der niedrigeren Nummer immer mit doppelter Frequenz „spielt", solange der NOP-Stecker eingesteckt ist. Es passiert nämlich folgendes, nachdem der RESET-Taster betätigt wurde: Die CPU sendet auf allen 16-A-Leitungen O-Pegel aus, während sie gleichzeitig MREQ und RD betätigt. Mit den Nullen auf den A-Leitungen will sie einem angeschlossenen Speicher signalisieren, daß sie mit der Speicherstelle O etwas vorhat. Sie sehen schon, worauf das hinausläuft: Die A-Leitungen, das sind die Adreßleitungen, mit welchen die CPU durch Aussenden einer 16stelligen Binärzahl bestimmt, welche Speicherstelle angesprochen werden soll.

Betrug

Mit dem eingesteckten Nichts-tu-Stecker wurde die CPU deshalb betrogen, weil ihr nur Speicher vorgetäuscht wurde. Nach dem Aussenden der Adresse und dem Betätigen der beiden Steuersignale MREQ und RD erwartet die CPU mit blindem Vertrauen, daß jetzt der Speicher mit dem Inhalt der angesprochenen Speicherzelle antwortet. Sie kann sich nicht vorstellen, daß nur ein so merkwürdiger Stecker die Pegel auf den Datenleitungen verursacht hat. Sie interpretiert also das Angebotene als Befehl, wie schon geschildert, und führt ihn durch. Die CPU ist nun so aufgebaut, daß sie nach der Durchführung des NOP-Befehls erneut einen Befehl holen möchte. Dazu sendet sie jetzt auf den Adressen-Leitungen die nächste Speicherzellennummer aus, eine Eins in diesem Fall.

Befehlszyklen

Bevor wir genauer den Speicher betrachten, sei also betont, daß die CPU im Experiment aus dem vorigen Kapitel durch den Reset-Impuls zunächst intern in einen Zustand gebracht wird, von dem aus sie die Adresse 0 auf den Adreßleitungen ausgibt und die kommende Antwort als Befehl interpretiert. Nach der Absolvierung des ersten Befehles, der diesmal ein NOP ist, sendet sie die Adresse 1 aus und erwartet wieder einen Befehl, der wegen des Steckers wieder ein NOP ist. Nach diesem sendet die CPU, so ist sie nämlich konstruiert, die nächste Adresse aus, die 2., danach die 3. und so fort. Jedesmal täuscht der Nichts-tu-Stecker den Inhalt NOP vor

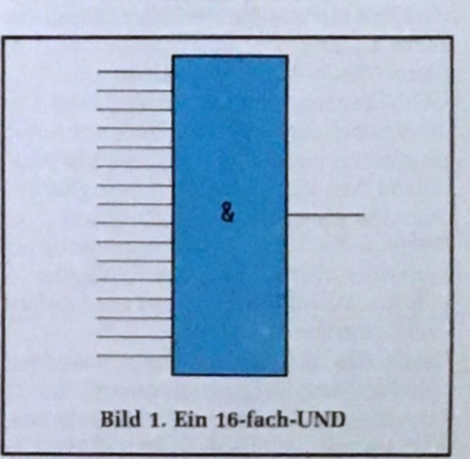


Bild 1. Ein 16-fach-UND

und die CPU tut ein paar Takte nichts. Das Spiel läuft durch, bis alle 16 Adressen-Leitungen 1 sind und hält auch dann nicht inne, denn die CPU ist so aufgebaut, daß sie dann wieder bei 0 anfängt. Das Experiment aus dem vorigen Kapitel würde also ewig laufen, wenn man nicht den Strom irgendwann abschalten würde. Die Adressen würden dabei immer wieder von 0 bis $2^{16}-1$, das ist 65 536–1, hochgezählt werden. Ebensoviele NOP-Befehle werden dabei absolviert. Vielleicht versuchen Sie einmal auszurechnen, wie lange die CPU für einen einzigen solcher Befehle benötigt.

Auswählen auf elektronisch

Die Z80-CPU ist also so konstruiert worden, daß sie einen Speicher von möglicherweise 65 536 Speicherzellen erwartet, von welchen sie in bestimmten Situationen genau eine Zelle auswählen möchte, um zu erfahren, was darin steht.

Die an die CPU angeschlossenen Speicherbausteine müssen so konstruiert sein, daß sie die Signale der CPU verstehen und entsprechend reagieren.

Ein wichtiges Detail dabei ist die Umsetzung der 16 Signale auf den Adreß-Leitungen, damit genau die gewünschte Speicherzelle angesprochen werden kann. Das Interessante ist, daß das mit den elementaren Logik-Bausteinen aus dem ersten Kapitel schon gelingt.
Das *Bild 1* zeigt dazu zunächst einen UND-Baustein mit 16 Eingängen. Gewiß zunächst ein Unikum. Dieser Baustein antwortet an seinem Ausgang genau dann mit einer 1, wenn alle seine 16 Eingänge auf 1 liegen. Ihn könnte man also dazu benutzen, genau diejenige Speicherzelle zu aktivieren, die die Nummer 65 535 trägt, denn wenn eine oder mehrere Adreßleitungen 0 führen, ist auch das UND nicht aktiv. Es reagiert wirklich nur auf die Zahl 65 535 und auf keine andere.
Setzen Sie jetzt gedanklich vor alle UND-Eingänge einen Inverter. Diese neue Schaltung ist genau dann am Ausgang aktiv, wenn alle Eingänge 0-Signal führen. Mit solch einer Schaltung kann man also die nullte Zelle anwählen.
Wenn beide Schaltungen gleichzeitig an denselben Adressenleitungen hängen würden, könnten also schon zwei Zellen exakt angesteuert werden.
Lassen Sie jetzt den Inverter, der an der Adreßleitung A0 hängt, aus der Schaltung weg, die die Null auswählen kann. Alle anderen behalten Sie bei. Diese Schaltung meldet sich genau dann, wenn die Binärzahl 1 auf den Adreßleitungen ausgesandt wird.
Wenn Sie den Inverter weglassen, der von Adreßleitung A1 bedient wird und den bei A0 ebenfalls, dann meldet sich diese Schaltung genau bei 3.

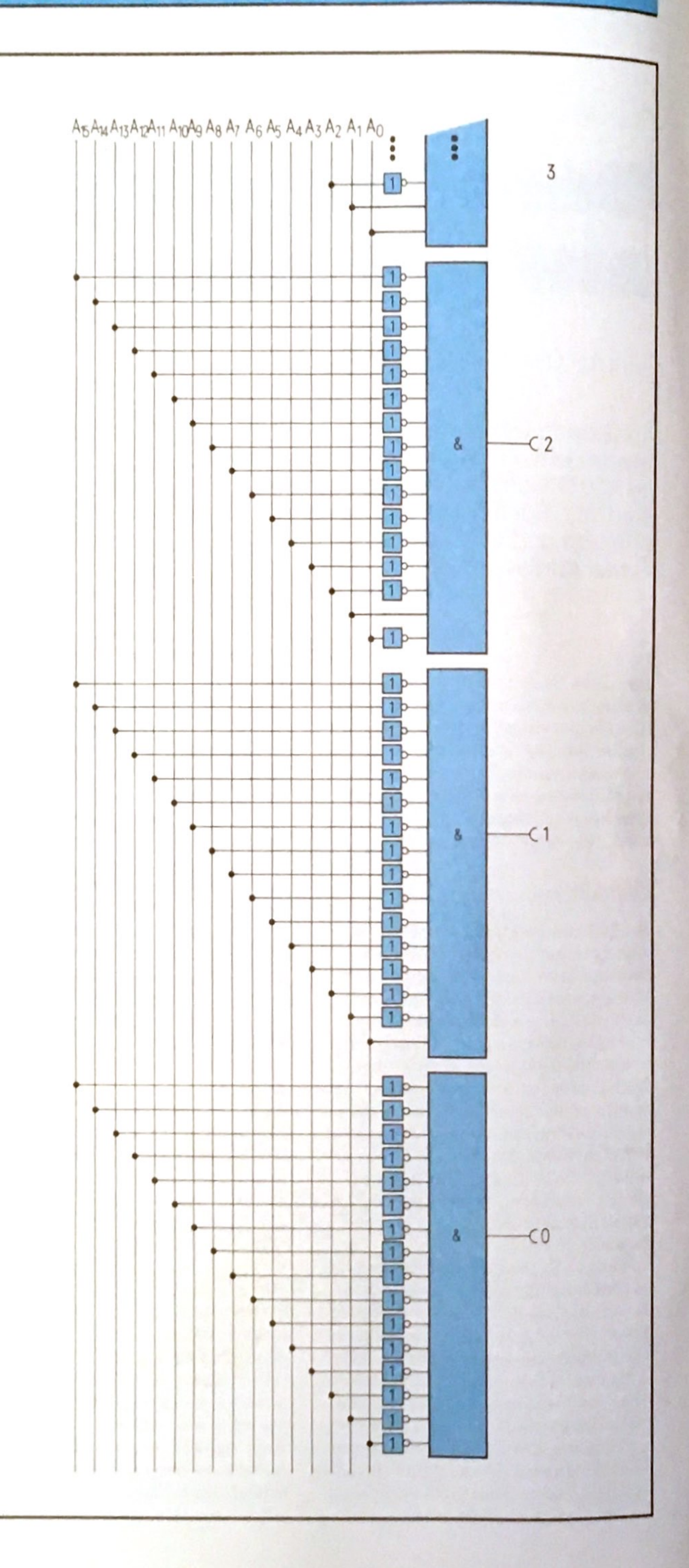


Bild 2. Auf 16 Adreßleitungen können 65 536 verschiedene Bitmuster erscheinen. Jedes Bitmuster soll genau ein Decoder-UND aktivieren. Das gelingt, wenn man 65 536 UNDs hernimmt und an deren Eingang entsprechend dem Bitmuster, bei dem es sich jeweils melden soll, Inverter verteilt. Und zwar kommt genau dorthin ein Inverter, wo das Bitmuster eine 0 zeigt. Entsprechend gibt es 16 Inverter beim „Null-Detektor", 15 beim „1-Detektor", wobei der an der Stelle A0 fehlt, ebenfalls 15 Inverter beim „2-Detektor", wobei der an der Stelle A1 fehlt und so weiter

Ganz allgemein: Wenn Sie eine Schaltung bauen wollen, die sich genau bei einer bestimmten Binärzahl meldet, dann nehmen Sie ein UND mit so vielen Eingängen, wie die Binärzahl Stellen besitzt, und setzen genau dort Inverter vor den UND-Eingang, wo die Binärzahl an der entsprechenden Stelle 0 stehen hat. Bei den Stellen mit 1 kommt kein Inverter davor. Genau bei dem vorgegebenen Bitmuster meldet sich dann die Schaltung.

Die Decodierung

Stellen Sie sich jetzt eine Schaltung vor, in der 65536mal 16-Fach-UNDs, nach dem vorhergegangenen Schema mit Invertern versehen, gleichzeitig an den 16 Adreßleitungen des Z80 hängen (*Bild 2*). Jedesmal, wenn der Z80 eine Binärzahl auf seine Adreßleitungen legt, meldet sich dann genau das eine UND an seinem Ausgang, das die Inverter an seinen Eingängen entsprechend verteilt hat. Man sagt, daß die eben vorgeschlagene Schaltung die Adressen des Z80 decodieren kann. Sie besitzt 65536 Ausgänge, von welchen jeweils genau der aktiv ist, dessen Nummer als Binärzahl angeliefert wird. Mit dem Ausgangssignal könnte man also genau die gewünschte Speicherzelle freischalten. Diese Riesenschaltung ist allerdings so nirgendwo in einem Speicherbaustein realisiert, weil sie viel Aufwand bedeuten würde.

Im Bausatz für die SBC2-Baugruppe ist aber doch auch genau solch ein Decoderbaustein beigelegt, weil er an bestimmter Stelle noch benötigt werden wird. Genauso heißt hier, daß er nach dem geschilderten Prinzip intern aufgebaut ist, allerdings nur drei „Adressenleitungen" nach außen geführt hat und entsprechend nur 8 Ausgänge besitzt, die jeweils aktiv werden, wenn das entsprechende Bitmuster, also die entsprechende dreistellige Binärzahl an seinen dafür vorgesehenen Eingängen anliegt (*Bild 3*). Er besitzt noch eine weitere Besonderheit: Den acht dreistelligen UNDs in seinem Inneren ist jeweils ein Inverter nachgeschaltet, so daß ein Ausgang mit 1 inaktiv ist, während ein Ausgang mit 0 anzeigt, daß seine „Zahl" an den Eingängen anlag. Und diese UNDs besitzen noch einen vierten Eingang, über den sie gesperrt werden können. *Bild 3* zeigt sein Schaltbild und seine Funktionstabelle. Daran kann man ablesen, daß neben den drei Eingängen A, B, C noch Steuereingänge G_1, G_{2A}, G_{2B} existieren, von welchen jeder die Ausgänge

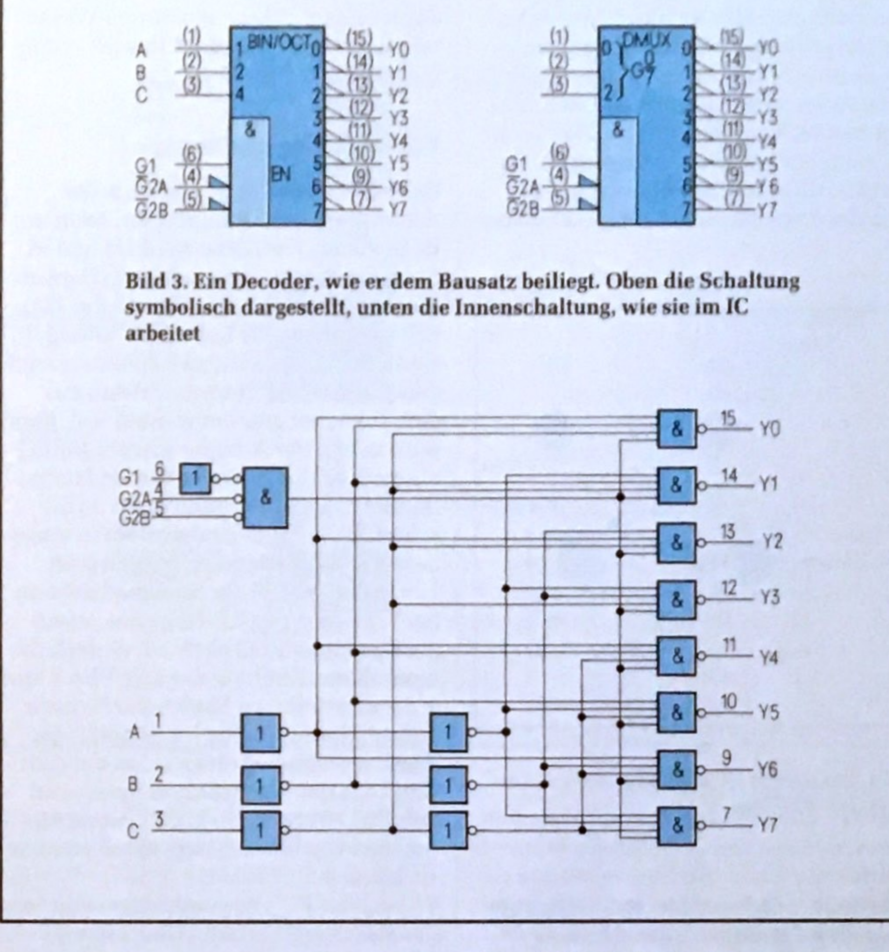


Bild 3. Ein Decoder, wie er dem Bausatz beiliegt. Oben die Schaltung symbolisch dargestellt, unten die Innenschaltung, wie sie im IC arbeitet

alle auf 1 bringen kann, wenn das entsprechende Potential an ihm liegt – und zwar unabhängig davon, was dann an den anderen Eingängen liegt.
Es gibt noch eine Besonderheit: Nicht bei jedem UND sind hier die Inverter passend verteilt, sondern nach den Eingängen A, B, C werden das invertierte A, B, C und das nichtinvertierte Signal bereitgestellt und die Eingänge der UNDs werden passend an das invertierte oder nichtinvertierte Signal gelegt. Das spart sehr viele Inverter. Bitte betrachten Sie Bild 3 genau. Solche Schaltungen sind wichtig.

Ins Innere der Speicherbausteine

Es gibt verwirrend viele Speicherbausteine, sowohl, was deren interne Funktionsweise als auch deren technologischen Aufbau betrifft. Im Prinzip aber wird bei allen Bausteinen zunächst eine Adresse decodiert und dann je nach Steuersignal entweder ein Inhalt ausgegeben oder ein neuer Inhalt eingegeben. Stellen Sie sich vor, daß jeder Ausgang eines Decoder-UNDs ein Relais bedient, das acht Kontakte besitzt und damit acht Bit-Leitungen gleichzeitig nach außen durchschalten kann. Im Inneren des Speichers sitzen nun noch acht Vorrichtungen bei jedem der Relais, die jeweils 0- oder 1-Pegel auf die Bitleitungen legen können. Beispielsweise könnten das acht Schalter sein, die ein oder aus sind. Wenn also am Decodereingang eine Binärzahl anliegt, dann wird das entsprechende Decoder-UND aktiviert und so der Zustand der dahinterliegenden Speicherzelle, also das Muster der dort eingestellten 0- oder 1-Pegel, durch das Relais auf die Ausgangsleitungen durchgeschaltet. Natürlich ist das eben Gesagte heute nicht mehr modern, zeigt aber exakt das Funktionsprinzip. Im Inneren der Speicherbausteine gibt es keine Relais, sondern nur noch Transistoren und manchmal auch kleine Kondensatoren. Sowohl die Relais als auch die Schalter, mit welchen die Bitmuster erzeugt wurden, sind aus solchen Transistoren nachgebildet.

Ein Flipflop

Damit Sie sich ein bißchen vorstellen können, wie in modernen Halbleiter-

speichern einzelne Bits abgespeichert werden können, sei folgendes Experiment gemacht (*Bild 4*): Sie nehmen die zweifache Treiberschaltung aus dem ersten Kapitel und führen da einen 10.-kΩ-Widerstand vom Ausgang auf den Eingang zurück. Der Effekt ist, daß bei nicht leuchtender Lampe am Ausgang hohes Potential über den Widerstand auf die Basis des Transistors am Eingang gelangt

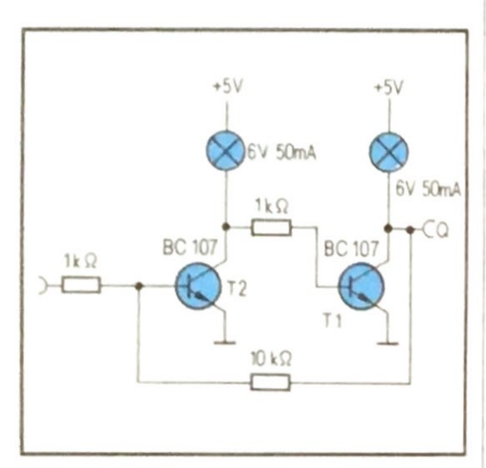


Bild 4. Ein positiv rückgekoppelter Verstärker ist ein Flipflop

und diesen geöffnet hält, was weiter bewirkt, daß Transistor 1 geschlossen bleibt, weil er keinen Basisstrom bekommt. Tippt man nun mit dem freien Pin des ebenfalls am Eingang von T2 liegenden 1-kΩ-Widerstandes an Masse, dann wird wegen des Widerstandsverhältnisses von 1 zu 10 das hohe Potential an der Basis von T2 gegen 0 V gezogen, der Transistor bekommt an seiner Basis nicht mehr genug Steuerstrom, sein Kollektorstrom verringert sich ebenfalls, weshalb auch das Potential dort ansteigt und das wiederum bewirkt, daß über den dort zur Basis von T1 abzweigenden 1-kΩ-Widerstand Strom fließt, der T1 öffnet. Ergebnis: Das einmal Tippen an 0 V ist dauerhaft als Stromfluß durch Transistor T1 gespeichert. Umgekehrt kann man mit der Steuerelektrode am Eingang an 5 V antippen und schon öffnet T2, während T1 schließt, weil er keinen Basisstrom mehr bekommt. Auch nach dem Tippen bleibt die Schaltung in der jeweiligen Lage. Also zusammengefaßt: Diese Schaltung kann sich merken, ob mit dem Steuereingang zuletzt an 0 V getippt wurde oder an 5 V. Eine solche Schaltung wurde von den Ingenieuren Flipflop getauft, nach dem Geräusch, das ein angeschlossener Lautsprecher macht, wenn umgeschaltet wird. Betrachten Sie diese Schaltung ein wenig mit Ehrfurcht. Neben dem Decoder ist sie (mit ihren modernen Varianten) wichtigstes Element in der Computertechnik.

Schreiben und Lesen

Das Flipflop selbst ist also eine der wichtigsten Schaltungstypen, denn es ist in vielen Varianten in CPUs und in Speicherbausteinen eingebaut. Denken Sie jetzt wieder an die Signale der Z80-CPU. Da gab es die Leitungen MREQ, RD und WR. Diese Leitungen führen je nach Absicht der CPU Impulse. Wenn aus dem Speicher gelesen werden soll, dann wird neben der Adresse sowohl MREQ als auch RD aktiv. Diese beiden letztgenannten Signale werden nun von oft schon in den Speicherbausteinen eingebauten Freigabelogiken ausgewertet. Und zwar so, daß der Speicherbaustein bei Vorliegen des RD-Impulses, wenn gleichzeitig MREQ aktiv ist, einfach die momentane Stellung der acht Flip-Flops einer angewählten Speicherzelle nach außen „durchschaltet". Das heißt, der Pegel am Ausgang eines jeden der acht Flipflops des angewählten Bytes wird auf die Leitungen D0 bis D7 gebracht, um dort von der CPU registriert werden zu können.
Wenn die CPU etwas schreiben möchte, dann aktiviert sie den WR-Ausgang, während RD inaktiv bleibt. In den Speicherbausteinen wird dieses Signal so ausgewertet, daß die auf den acht Datenleitungen von der CPU erzeugten Pegel jetzt genau den acht Eingangssteuerleitungen der acht von den Adressenleitungen angewählten Flipflops zugeführt wird, die genau so ihre Lage danach richten, wie das eine handgemachte Flipflop im Experiment, und diese Lage auch dann beibehalten, wenn alle Steuerimpulse vorbei sind.
Es sei nochmals betont, daß alle Dinge, die bisher geschildert wurden, eher die Funktionsprinzipien schildern, als den wirklichen Aufbau von Speicherzellen, denn dort haben sich im Zuge der Entwicklung der Technik noch höchst raffinierte Varianten herausgebildet, die nicht so leicht erkennen lassen, auf welch einfachen Tatsachen alles beruht.

Die verschiedenen Speichertypen

Bild 6 zeigt eine Reihe von Speichertypen. Wir betrachten zunächst den Speicher mit der Beschriftung RAM. RAM bedeutet, Random Access Memory oder auf deutsch: Speicher mit wahlfreiem Zugriff. Das soll anzeigen, daß man Informationen in diesem Speicher einschreiben und auch wieder auslesen kann, genauso, wie vorhin geschildert. Solche ICs besitzen einen Eingang, der mit „Aktivieren" beschriftet ist. Liegt dort ein Signal an, so wird der Speicher in Arbeitsbereitschaft versetzt. Erst jetzt achtet er auf Signale an seinen anderen Anschlüssen.
Dann gibt es einen Dateneingang. Der muß nicht immer 8 Bit breit sein, meist ist er sogar nur 1 Bit breit. Dort wird die Information angelegt, also zum Beispiel, ob ein 1-Signal oder ein 0-Signal gespeichert werden soll. Ferner findet man einen Eingang „Schreiben". Erscheint dort ein Puls, so wird die Information am Dateneingang in den Speicher übernommen. Wenn man Informationen auslesen will, so legt man einen Puls an den Eingang „Lesen". Am Datenausgang erscheint dann die Information.
Dann gibt es noch einen Eingang mit der Beschriftung „Adresse". Dahinter verbergen sich die Decoderelemente. Bei den Speicherbausteinen in der Wirklichkeit werden die Speicher-Flipflops in einer Anzahl von Spalten angeordnet.

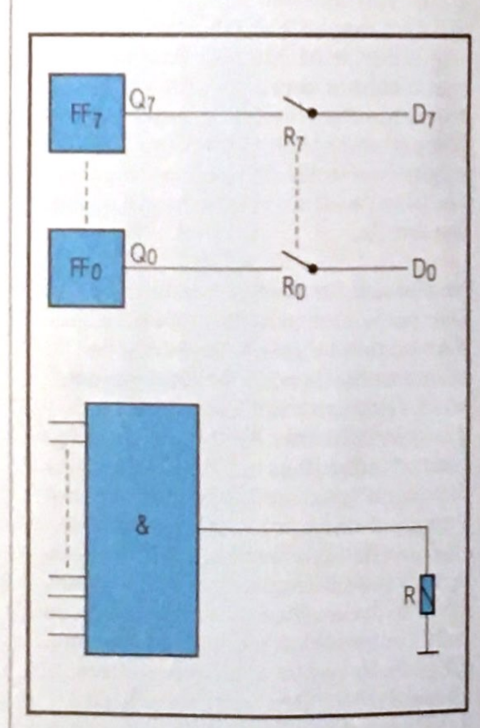


Bild 5. Acht Kontakte schalten ein Byte auf den Datenbus

Um eine Speicherzelle zu „adressieren", werden intern eine Spalte und darin eine Reihe angesprochen (engl. columns and rows). Eine einzelne Speicherzelle wird nur aktiviert, wenn die Spaltenleitung und die Zeilenleitung aktiviert

mikrocomputer schritt für schritt

sind. Durch die Matrixanordnung wird die Zahl der benötigten UNDs im Decoder von n auf $2 \times \sqrt{n}$ reduziert. Beispiel: Gegeben sind 1000 Speicherzellen, dann sind also 20 UNDs nötig, somit je 10 Reihen und 10 Spalten. Denn $10 \times 10 = 100$.
Ein anderes Beispiel: Ein Speicher faßt 65 536 Bit. Wieviele Adreßleitungen braucht man? Den Speicher kann man in 256 × 256 Speicherzellen aufteilen, also hat man je 256 UNDs bei den Spalten und bei den Reihen zu codieren. Dazu benötigt man je 8 Dualstellen, gesamt also 16 Dualstellen. Mit 16 Adreßleitungen kann man also eine Speicherzelle aus diesem Speicher auswählen.

Von EPROMs und ROMs

In *Bild 6* sind noch andere Speicherbausteine gezeigt. Da gibt es zum Beispiel sogenannte ROMs. Ihnen fehlt der Schreibeingang und der Dateneingang. Man kann aus diesen Bausteinen nur Daten lesen. ROM ist die Abkürzung für Read Only Memory, zu deutsch „Nur-Lese-Speicher".

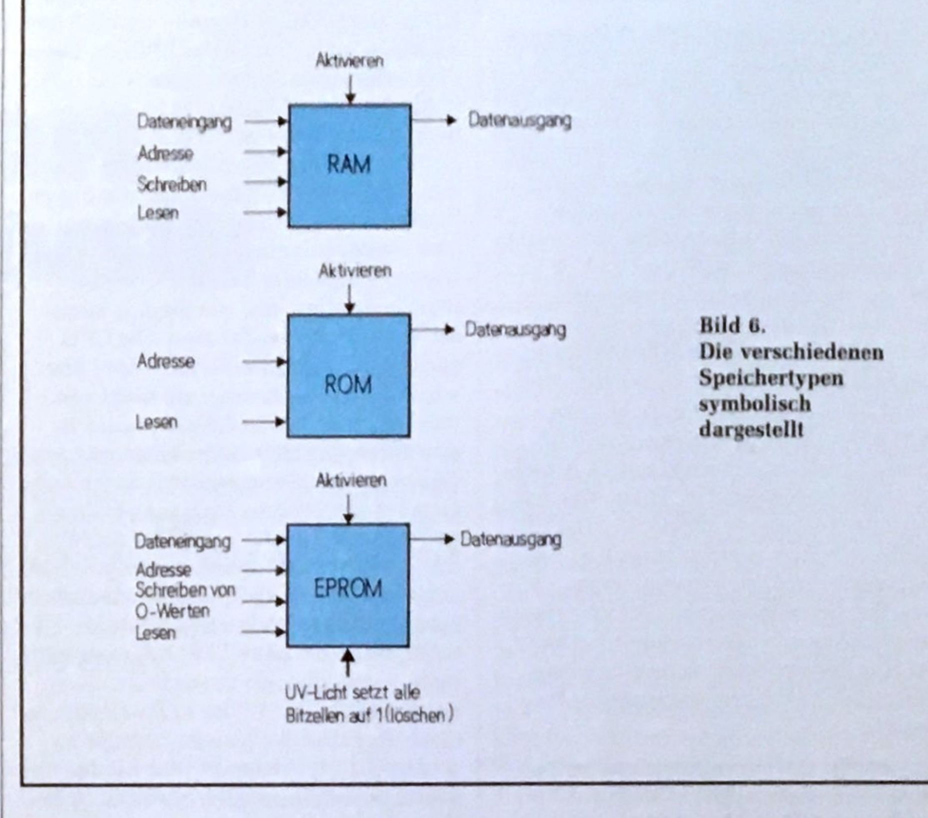


Bild 6. Die verschiedenen Speichertypen symbolisch dargestellt

Bei ROMs geschieht der Einspeichervorgang schon bei Herstellung der integrierten Bausteine. Durch einen metallischen Aufdampfungsprozeß werden die Informationen (oft durch gezieltes Schließen oder Offenlassen einer Bitleitung wie bei einem Schalter) fest vorgegeben. ROMs werden immer dann verwendet, wenn zum einen die Informationen dauerhaft sein sollen, und zum anderen sehr große Stückzahlen verwendet werden sollen, denn der metallische Aufdampfungsprozeß ist teuer, wollte man nur einzelne ICs herstellen. Wenn man Einzelstücke mit festem Inhalt benötigt, so verwendet man die sogenannten EPROMs, Eraseable Programmable Read Only Memories. Auf deutsch: Löschbare Programmierbare Nur-Lese-Speicher.

Diese EPROMs haben wieder einen Schreibeingang, jedoch kann man damit nur 0-Signale einschreiben, wenn also im Speicher schon ein 0-Signal gespeichert war, so kann man daraus kein 1-Signal mehr machen. Dazu gibt es aber im Deckel des EPROMs ein Quarzfenster. Wenn man das EPROM durch dieses Quarzfenster mit UV-Strahlen belichtet, so werden alle Speicher-Zellen in den 1-Zustand überführt. Dieser Prozeß dauert etwa eine viertel Stunde. EPROMs sind also empfindlich gegenüber Licht. Tageslicht ist in der Lage, EPROMs innerhalb weniger Stunden zu löschen, also 1-Signale einzuspeichern.

Wenn man EPROMs verwendet, sollte man also besser den Quarzdeckel mit einem Aufkleber versehen und so das IC vor Licht schützen.
Das Einschreiben von Informationen in EPROMs hat noch eine Besonderheit. Es dauert relativ lange. Während man Daten im Bereich 200 ns (Nanosekunden, $1 \text{ ns} = \frac{1}{1000000000} \text{ s}$) auslesen kann, braucht man 50 ms (Millisekunden, $1 \text{ ms} = \frac{1}{1000} \text{ s}$) zum Einschreiben der Daten in eine einzelne Speicherzelle.
Daher verwendet man EPROMs als ROM-Ersatz, wenn es darum geht, Daten oder Programme dauerhaft zu speichern. Leider kann man EPROMs nicht beliebig oft löschen, sie lassen sich irgendwann einmal nicht mehr programmieren.
Wie wird die Information in einem EPROM gespeichert?
Das ist ganz trickreich. Im EPROM befinden sich speziell gebaute Kondensatoren. Auf diese Kondensatoren wird beim Einschreiben eine Ladung gebracht. Diese Ladung bestimmt, ob ein 1-Signal gespeichert ist (keine Ladung da) oder ein 0-Signal (Ladung da). Durch die UV-Strahlung wird die Ladung wieder entfernt und damit der Speicher mit 1-Werten belegt.

Dynamische RAMs

Bei RAM-Bausteinen gibt es Speicher, die mit Kondensatoren arbeiten. Man nennt sie dynamische Speicher. Da bedeutet ein geladener Kondensator 1 und ein ungeladener 0. Die Kondensatoren dieser Speicher verlieren aber nach etwa 2 bis 4 ms ihre Informationen. Daher müssen die Daten vor Ablauf dieser Zeit immer wieder neu aufgefrischt werden. Hilfsschaltungen für diesen Zweck sind dabei meist mit in das IC eingebaut, man nennt den Vorgang englisch „Refresh".

Allerdings muß der Refresh von außen angestoßen werden und es muß dafür meist auch eine Speicheradresse mitgegeben werden (zumindest von einer Spalte oder Reihe), deren Inhalt dann automatisch aufgefrischt wird. Diese dynamischen Speicher sind im allgemeinen mit einer viermal so hohen Kapazität verfügbar als die sogenannten statistischen Speicher. Beim dynamischen Speicher ist nämlich nur ein Transistor zur Abfrage der Kondensatoren nötig, während bei den statischen Speichern, die mit Flipflops arbeiten, zu den zwei Flipflop-Transistoren nochmals zwei Transistoren kommen, die das Flipflop im Betrieb bedienen. Dynamische Speicher gibt es zur Zeit mit 262 144 Bit, (im Labor 1 048 576) auf einem Chip, während bei statischen Speichern derzeit bis zu 65 536 Speicherzelle auf einem Chip verfügbar sind.

Ein neuer Schritt beim Aufbau

Zum Abschluß wird die SBC2-Baugruppe weiter aufgebaut. Dazu werden die beiden Decoder 74LS138 in die zwei 16-poligen Fassungen gesteckt (Wo ist Pin1?). Der eine Decoder übernimmt die Aufgabe der Decodierung (Zuordnung einer Adresse zu einem Speicherbaustein) für die RAM-Bausteine, die später in die Fassungen 2 (IC8) und 3 (IC9) gesteckt werden und der andere für die EPROMs, die in die Fassungen 0 (IC6) und 1 (IC7) kommen. Es ist nämlich so, daß zum Beispiel die verwendeten statischen Speicherbausteine nicht volle 65 536 Byte fassen, sondern nur 2048. Deshalb besitzen sie auch nur 11 Adreßleitungen herausgeführt. Diese sind mit den Adreßleitungen A0 bis A10 der CPU verbunden. Um nun Speicher ICs „aneinanderreihen" zu können, haben sie den Aktivierungseingang, der von einem externen Decoder bedient werden kann. Der hier verwendete Decoderbaustein könnte bis zu 8 Speicherbausteine bedienen.

Wir wollen die Decoder einmal testen. Dazu wird nochmal der Nichts-tu-Stekker benötigt. Er kommt in Fassung 0 (IC16). Nach dem Einschalten kann man nun am RAM-Decoder und am ROM-Decoder messen. Pin 15, 14, 13, 12, 11, 10, 9 und 7 sind die Ausgänge der Decoder. Dort müssen Pulse sichtbar werden. Wenn man den Prüfstift verwendet, so kann man dies auch deutlich erkennen. IC4 ist der EPROM-Decoder und IC5 der RAM-Decoder. Pin 15 des EPROM-Decoders zeigt etwas andere Signale als die restlichen Anschlüsse, das ist jedoch normal und hängt mit dem Aufbau des Z80 zusammen. Wenn man beide Decoder vergleicht, sieht man, daß die Signale recht ähnlich aussehen. Man kann hier einmal mit einem Zweikanal-Oszilloskop-Vergleiche durchführen und wird feststellen, daß die Signale zueinander zeitlich versetzt sind. Die CPU spricht ja nacheinander alle Speicheradressen an (auch wenn sie nicht vorhanden sind). Seine Eingänge sind an den höherwertigen Adreßleitungen des Z80 befestigt. Wenn man mit dem Skop genauer hinsieht, so kann man feststellen, daß die Auswahl eines bestimmten Speicherbausteines eine bestimmte Zeit lang durch viele kleine Pulse wiederholt geschieht. Genau sind es 2048 beim RAM und 4096 beim EPROM. Aber bitte nicht versuchen, sie abzuzählen. Ausnahme bildet Pin 15 des ROM-Decoders, denn dort sind noch mehr Impulse zu finden als nur die Lese-Pulse für den speicher. Und hier noch etwas zu Speicherorganisationen.

Unser RAM-Speicher hat eine Kapazität von 16 384 Bit. Hier sind die Flip-Flops nicht in einer einfachen 128×128-Matrix angeordnet, sondern als 64×32×8-Matrix. Das bedeutet, es sind acht Ebenen vorhanden. In diesen Ebenen werden jeweils bei Anlegen einer Adresse je genau ein Flipflop angewählt, so daß auf 8 Datenleitungen ein Byte zur Verfügung steht.

Weiterhin fällt im Schaltbild vielleicht auf, daß es keine getrennten Daten-Ein- und Ausgänge gibt. Dies ist auch nicht unbedingt nötig, denn bei diesen Speichern wird immer nur entweder gelesen oder geschrieben, so daß man die Leitungen für Ein- und Ausgang gemeinsam benutzen kann. Wir haben also auch hier „bidirektionale" Leitungen, wie wir sie schon beim Z80 kennengelernt haben.

Zum Nachdenken

1. Was ist der Unterschied zwischen ROM und EPROM?

2. Warum kann man ein EPROM nicht als RAM verwenden und welche Nachteile hätte das?

3. Wieviel Adreßleitungen besitzt der Speicher 6116, den wir auf der SBC2-Baugruppe verwenden? Er besitzt eine Organisation von 2048 × 8 Speicherzellen, und 8 Datenleitungen sind direkt herausgeführt. (Lösung siehe Schaltplan, durch Abzählen der Leitungen A0, A1, A2...)

```
8800:
 MEHRKREISE:=$
  21 #300.W
  22 8900.W
  21 #120.W
  CD SCHLEIFE
  21 #36.W
  CD SCHLEIFE
  2A 8900.W
  CD SCHR16TEL
  21 #10.W
  CD DREHE
  CD ENDSCHLEIFE
  2A 8900.W
  11 -#5.W
  19
  22 8900.W
  CD ENDSCHLEIFE
  C9
```


Rolf-Dieter Klein

Ein EPROM macht Musik

Mikroelektronik, Folge 8

Es ist schwierig, einen Mikrocomputer wie den SBC2, der ja schon fast fertig aufgebaut ist, etwas tun zu lassen, ohne genau zu wissen, wie man ihn programmiert. Er kann nur nach einem Programm, einer Folge von mehr oder manchmal auch weniger sinnvollen Befehlen, arbeiten. Aber für Sie wird das doch leicht, denn es gibt fertig programmierte EPROMs, die Sie nur einstekken müssen – und schon tut Ihr Z80 etwas.

Musik scheint zunächst mit Mikrocomputern nichts zu tun zu haben. Musik, von Computern dargeboten, ist aber das beste Beispiel, wie sich Computer und Mikrocomputer mehr und mehr als Medium für Künstler und Kunstbetrachter nützlich machen. Das gelingt, weil die Computer trotz ihrer digitalen Struktur, die Sie in Form der Bits und Bytes schon kennengelernt haben, aufgrund ihrer Rechenkraft so viele Vorgänge aus der Natur elektronisch in ihrem Inneren nachahmen können, daß die Fantasie nicht reicht, sich alles, was möglich ist, auszudenken. Musik gehört dazu.

Musik digital

Wer schon einmal ein Radio auseinandergenommen hat, der weiß, daß die Musik daraus von einem Lautsprecher erzeugt wird, der der umgebenden Luft die Töne aufprägt, indem er sich schnell hin und her bewegt. Wenn eine Geige erklingen soll, bewegt er sich ziemlich genau so, daß alles, was die Geigensaite an Luftschwingungen erzeugt, auch von ihm der umgebenden Luft aufgeprägt wird. Dabei muß der Lautsprecher ebensooft hin und her schwingen, wie es die Geigensaite tut. Man kann nun in langen Zahlenkolonnen einmal aufschreiben, um wieviel tausendstel mm sich gerade der Lautsprecher aus seiner Mittellage entfernt hat, wobei man diese Messung zum Beispiel 20 000mal pro Sekunde durchführt. 1 mm Auslenkung entspräche dann der Zahl 1000. Es gibt Geräte, die können solche Messungen durchführen. Meistens liefern diese sogenannten Analog-Digital-Wandler ihr Meßergebnis in Form von Binärzahlen ab. Andererseits gibt es auch Wandler, die auf der einen Seite eine Binärzahl erwarten und auf der anderen Seite exakt den Strom oder die Spannung abliefern, die der eingegebenen Binärzahl entspricht. Was zum Beispiel einen Lautsprecher um soviel tausendstel eines Millimeters auslenken würde. Wenn man die vorhin gewonnene Meßreihe nehmen würde und deren Meßwerte exakt so schnell hintereinander über einen solchen sogenannten Digital-Analog-Wandler jagen würde, wie bei der Aufnahme, dann erklänge der Geigenton wieder ganz exakt so, wie vor der Wandlung in Zahlen. Die modernen Digital-Schallplatten-Spieler arbeiten nach diesem Prinzip. Auf der kleinen silbrigen Scheibe sind Binärzahlen in Form von Lochmustern eingebrannt, die exakt den momentanen Luftdruckwerten entsprechen, die zum Beispiel eine Big Band einmal erzeugt hat. Der Plattenspieler liest die gewaltige Zahlenkolonne und präsentiert sie als Musik.

Musik aus der SBC2

Ob die Zahlenkolonnen aus Meßreihen gewonnen wurden, oder ob diese Zahlenkolonnen, die über einem Wandler dem Lautsprecher zugeführt werden, von einem Computer nach irgendeinem Programm erzeugt werden, das ist dem Lautsprecher egal. Er macht einen schönen oder häßlichen Ton daraus – ganz nach der Art der Zahlenkolonne. Manche Musiker sind ganz fasziniert von diesen Möglichkeiten.

Die Musik, die wir jetzt mit der SBC2 erzeugen wollen, wird ebenfalls von einem Programm erzeugt. Es wird aber kein so teurer Digital-Analogwandler benutzt. Das Programm schaltet einfach den $\overline{IORQ}$-Ausgang der CPU so blitzschnell ein und aus, daß das Ergebnis recht hörsame Musik ist. Nach dem, was Sie bisher kennengelernt haben, könnten Sie vermuten, daß man nur genügend viele NOP-Befehle und OUT-Befehle geeignet kombinieren müßte, damit Musik erklingt. So ginge es zwar, aber der Z80 besitzt sehr viele weitere Befehle, die zwar noch nicht besprochen sind, die aber viel wirksamer beim Musikmachen helfen können. Diese Befehle sind im EPROM benutzt. Später werden Sie auch diese Befehle kennenlernen. In diesem und in weiteren Kapiteln geht es aber nicht um diese Befehle, sondern Sie sollen eher kennenlernen, wie das Spiel der Signale eines Mikrocomputers intern verläuft und wie es gezielt nach außen gegeben werden kann.

Stecken Sie also das EPROM MUO in den Sockel 0 der SBC2-Karte, also da, wo sich früher der Nichtstu-Stecker befand. Dabei unbedingt auf die Lage der Markierung des EPROMs achten, denn wenn man es verkehrt herum einsteckt, wird es zerstört. Also: lieber nochmals genau kontrollieren. Das EPROM enthält eine Invention in F von Johann Sebastian Bach.

Nun den Rechner einschalten. An Pin 20, dem $\overline{IORQ}$-Pin der CPU wird gemessen. Dort erscheint ein Signalmuster ähnlich *Bild 1*. Das Muster ändert sich rhythmisch mit der Musik. Mit dem Prüfstift allein kann man das Signal ebenfalls erahnen. Die LEDs W1 und W2 leuchten beide, genauso wie die LED H. Die LED L bleibt dagegen fast ganz dun-

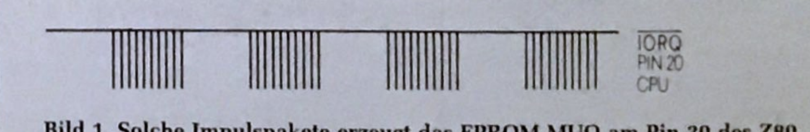


Bild 1. Solche Impulspakete erzeugt das EPROM MUO am Pin 20 des Z80

kel, denn der $\overline{\text{IORQ}}$-Pin führt sehr kurze Signale, die auf 0 V gehen.

Musik, mit Rechteck erzeugt

Wenn Sie mit dem Oszilloskop messen wollen, dann sollte die Zeitbasis auf etwa 1 ms/cm gestellt werden. Der Abschwächer auf 2 V/cm. Die Triggerung erfolgt automatisch.
Das Signal besteht aus periodisch wiederkehrenden Impulspaketen. *Bild 2* zeigt nun noch ein zweites Signal. Das soll das „Musiksignal" sein, das dem Lautsprecher angeboten wird. Es ist ein „Rechtecksignal", das das Sinussignal normaler Musik ersetzen soll.
Die Breite T ist die Periodendauer des Signals, das die Tonhöhe bestimmt. Durch Verändern des Abstandes der Pulspakete an Pin 20 kann man diese Periodendauer verändern und somit die Tonhöhe der erzeugten Schwingung. Daß wir keine richtige schöne Schwingung haben, stört nicht weiter, denn die Impulse des Impulspaketes liegen so dicht aneinander, daß sie keine neuen hörbaren Obertöne erzeugen.

Nun kann man einen Lautsprecher anschließen. Doch halt! Ein Lautsprecher besitzt einen Widerstand von 8 Ohm und weniger. Man darf solche geringen Widerstände nicht direkt an die Ausgänge der CPU anschließen, da die CPU nur sehr wenig Strom liefern kann.

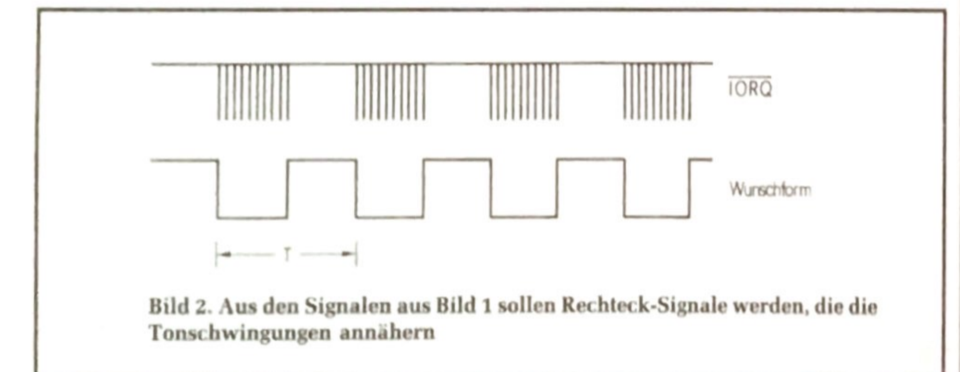


Bild 2. Aus den Signalen aus Bild 1 sollen Rechteck-Signale werden, die die Tonschwingungen annähern

Ein Verstärker

Wer ein Radio besitzt, kann jetzt einmal probieren, den Tonband-Eingang mit dem $\overline{\text{IORQ}}$-Ausgang der CPU zu verbinden. Doch die Musik wird noch nicht sehr laut sein.
Es sei jetzt ein eigener Verstärker gebaut. Dieser Verstärker hat eine Besonderheit. Da nur die Signale 0 und 1 auftreten, muß er nur zwischen diesen beiden Werten aus seinem Eingang unterscheiden können. Wir brauchen also einen digitalen Verstärker, wie er bereits am Anfang des Kurses aufgebaut wurde.

Wir bauen ihn neu mit einem der Leistungstransistoren TIP120 oder TIP110 auf. *Bild 3* zeigt die Schaltung. Es gibt aber noch ein Problem. Dazu betrachten wir nochmals das Oszilloskop. Diesmal wird mit einer Zeitablenkung von 2 µs (Mikrosekunden) gemessen. Man sieht, daß der $\overline{\text{IORQ}}$-Puls nur ca. 600 ns (Nanosekunden) breit ist (*Bild 4*). Ferner beträgt der Abstand zwischen zwei Pulsen 8 µs (Mikrosekunden). Das Signal liegt also die meiste Zeit auf 1. Wenn man dieses Signal an den Transistortreiber legt, so geschieht folgendes: Bei einem 1-Signal leitet der Transistor und der Lautsprecher steht unter Strom. Bei einem 0-Signal, das aber nur für 600 ns erscheint, leitet der Transistor nicht und der Lautsprecher ist für diese kurze Zeit stromlos. Die meiste Zeit fließt also Strom durch den Lautsprecher. Das ist weder energiesparend, noch gut für den Lautsprecher und wenn man es einmal ausprobiert, so stellt man fest, daß die Musik nur sehr schwach zu hören ist.

Ein Monoflop zur Impulsverlängerung

Dagegen kann man etwas tun. Zum einen müssen die Pulse breiter werden und dann sollten sie invertiert werden,

Bild 3. Das ist der Verstärker, der die Rechtecksignale zu Gehör bringt

das heißt, der Ruhepegel soll auf 0 V liegen und bei Aktivierung kurz auf 5 V (bzw. 2,4 bis 5 V) ansteigen.

Bei der Messung mit dem Oszilloskop stellt man übrigens fest, daß die Signale nicht so ideal aussehen, wie auf unseren Abbildungen. Dem 1-Signal-Teil sind lauter kleine Rechtecke überlagert, die ein wildes Muster bilden. Die Amplitude dieser Störung beträgt etwa 1 V. Das Gesamtsignal liegt aber stets im 1-Signal-Bereich von 2,4 V bis 5 V. Die Überlagerungen bewirken keine Störungen beim Betrieb. Sie treten bei allen Digitalschaltungen auf und rühren von Schaltvorgängen im Inneren der ICs her. Man kann sie nicht beseitigen. In *Bild 5* ist nun ein Vorschlag zur Signalverbesserung gemacht. Monoflop wurde schon auf der SBC2-Karte verwendet, um den Reset-Impuls zu erzeugen. Mit dem Kondensator C1 kann man die Breite des Pulses einstellen. Am Ausgang (Pin 6) des Monoflops ist der Treiber angeschlossen. Das Monoflop gibt auch ein Signal mit positivem, aktivem Pegel aus, so daß am Pin 6 das fertige Signal ansteht, das über den Widerstand R1 direkt an den Transistor T1 gelangt. Der Transistor T1 steuert den Lautsprecher an. Der Lautsprecher ist über einen Widerstand R2 mit 5 V verbunden. Dieser Widerstand dient zum Schutz vor Kurzschluß. Der Widerstand R3 liegt am Eingang Pin 5 des ICs 74 121 und hat die Aufgabe, ein 1-Signal an diesen Eingang zu legen. Man könnte den Widerstand auch weglassen und den Eingang direkt mit 5 V verbinden, jedoch sollte man dies bei TTL-Baustein-Eingängen nicht tun, da die Eingänge dann bei Überspannung (> 5,5 V) zerstört werden können. *Bild 6* zeigt die Signale. Immer wenn das $\overline{\text{IORQ}}$-Signal von 1 auf 0 abfällt, wird das Monoflop ausgelöst (getriggert). Bei uns ist die Zeitdauer auf ca. 18 µs eingestellt. Dann fällt das Monoflop wieder auf 0. Bei einem neuen Signalwechsel des $\overline{\text{IORQ}}$-Signals wird es wieder ausgelöst. Man sieht, daß der $\overline{\text{IORQ}}$-Puls jetzt sehr stark verbreitert ist. Dabei ist die neue Breite sogar größer als der Abstand der ursprünglichen Pulse.
Wenn man das Signal bei 1 ms/cm auf dem Oszilloskop ansieht, so erkennt man, daß nun das Signal die meiste Zeit den Wert 0 besitzt und nur für die Zeitdauer der Pulse längere Zeit den Wert 1 annimmt. Das ist aber genau das, was wir wollten.

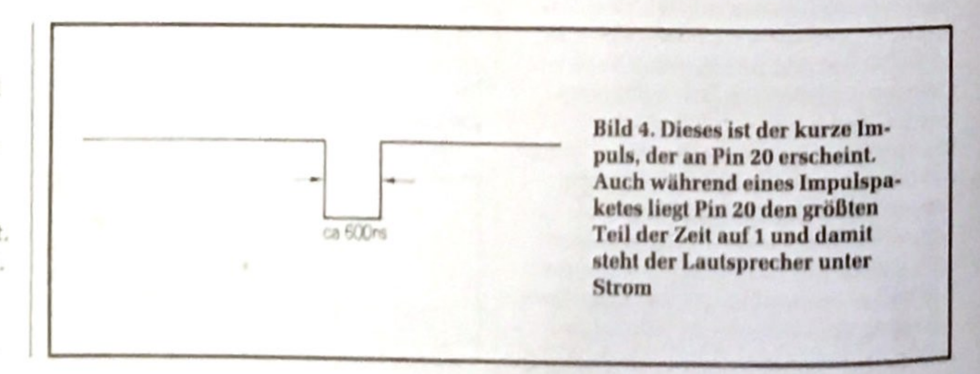


Bild 4. Dieses ist der kurze Impuls, der an Pin 20 erscheint. Auch während eines Impulspaketes liegt Pin 20 den größten Teil der Zeit auf 1 und damit steht der Lautsprecher unter Strom

Zum Aufbau

Wer Mut besitzt, kann alles auf einer Lochrasterplatte aufbauen und selbst verdrahten. Es gibt aber auch eine fertige Leiterplatte im Handel, die laut dort beigefügtem Bestückungsplan aufgebaut wird.
Bild 8 zeigt die Belegung des TIP120 (TIP110), für alle, die den Aufbau selbst wagen wollen. Hier noch ein Tip. Man kann die Schaltung mit isoliertem Kupferlackdraht verdrahten. Dieser sollte lötfähig sein und nicht zu dünn; ca. 0,3 mm Querschnitt sind gerade richtig. Man kann die Verdrahtung aber auch mit normaler Litze durchführen.
Bild 9 zeigt den Aufbau mit der vorgefertigten Leiterplatte. Die Baugruppe Musik wird an die Versorgungsspannung angeschlossen. Dazu kann man Leitungen verwenden oder die Grundplatte (BUS1). Das $\overline{\text{IORQ}}$-Signal verbindet man ausgehend von der SBC2-Karte mit dem Eingang der Musikschaltung.
Wenn nach dem Einschalten keine Musik am Lautsprecher erscheint, so beginnt jetzt die Fehlersuche.

Bild 5. Mit einem Monoflop, wie er bei der Reset-Schaltung schon benutzt wurde, wird der Impuls an Pin 20 verlängert

Nach dem Aufbau: Test

1. Kontrolle, ob das $\overline{\text{IORQ}}$-Signal noch vorhanden ist. Dazu kann man mit dem Oszilloskop oder mit dem Prüfstift das Signal verfolgen. Beim Prüfstift müssen die LEDs W1, W2 und H leuchten.
2. Messen an Pin 3 und 4 des ICs 74 121 auf der Musikschaltung. Dort muß dieses Signal ankommen.
3. Messen an Pin 6 des ICs 74 121. Dort muß sich mit dem Prüfstift ein ähnliches Bild wie oben ergeben; W1 und W2 sowie H leuchten, die LED L leuchtet jetzt ebenfalls fast genauso stark.
4. Transistor-Treiber. Am Kollektor des Transistors muß dieses Signal ebenfalls ankommen.
5. Nun bleibt nur noch der Lautsprecher. Er muß über den 47-Ω-Widerstand mit 5 V verbunden sein.

Wenn irgendwo das Signal ausbleibt, so liegt der Fehler aller Wahrscheinlichkeit zwischen dem Punkt, wo es noch auftrat und dem, wo es zum ersten Mal nicht mehr zu messen ist. Man nennt dieses Verfahren der Fehlersuche Signalverfolgung. Damit lassen sich Fehler meist recht schnell finden.
Man sollte auch die Versorgungsspannungen an den integrierten Schaltungen überprüfen. Eine vergessene Leitung, und meist funktioniert die Sache dann nicht mehr.
Das Musik-Programm kam ganz ohne RAM aus, da es sich nichts extern merken mußte. Im Bausatz befinden sich jedoch zwei RAM-Bausteine. Für die folgenden Versuche benötigt man eigentlich nur einen davon, doch schadet es nicht, wenn man schon beide Bausteine einbaut. Die RAMs kommen in die Fassungen 2 (IC 8) und 3 (IC 9) (siehe Bestückungsplan).

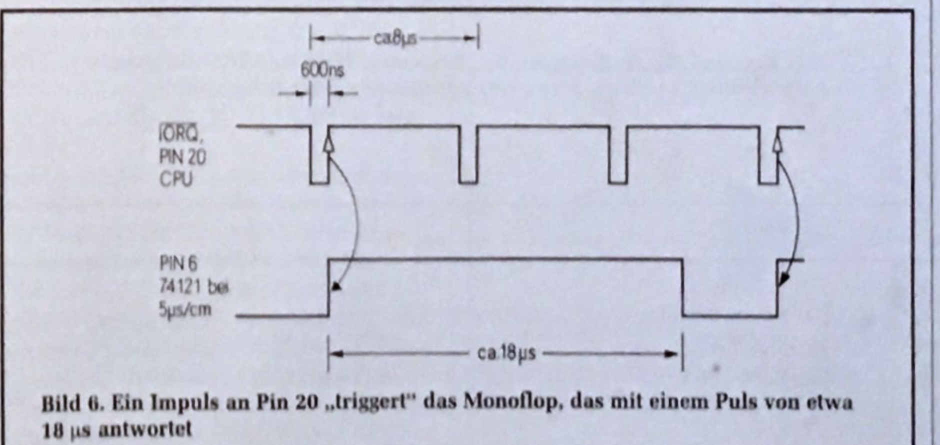


Bild 6. Ein Impuls an Pin 20 „triggert" das Monoflop, das mit einem Puls von etwa 18 µs antwortet

Speichertest und Riesenzahlen

Nun muß getestet werden, ob das RAM auch funktioniert. Dazu verwendet man ein sogenanntes Speichertestprogramm, wie die Profis in der Industrie. Man muß nämlich damit rechnen, daß nicht alle RAM-Bausteine funktionieren, die aus der Halbleiterfertigung kommen. Zum Beispiel durch Verunreinigungen der Silizium-Platten (Wafer genannt), die das Grundmaterial bilden. Die einzelnen

PIN 6
74121
bei 1 ms/cm

Bild 7. Das ist das Ausgangssignal des Monoflops

Speicherzellen eines RAMs kann man testen, indem man zunächst eine 0 in die Zelle einschreibt, dann prüft, ob die 0 angekommen ist und dann eine 1 in die Speicherzelle schreibt und nochmals prüft, ob die 1 auch gespeichert wurde. Das müßte man mit jeder einzelnen Speicherzelle tun, um den ganzen Speicher zu testen.

In der Praxis genügt solch ein Test nicht. Man stelle sich zwei benachbarte Zellen vor, die elektrisch miteinander kurzgeschlossen sind. Dann kann man diesen Fehler mit dem Test nicht finden, denn beim Einschreiben einer 0 in die eine Zelle käme das Datum auch in die andere Zelle und bei einer 1 ebenfalls. Der Test „in eine Zeile schreiben und gleich auszulesen" ist also nicht besonders gut. Rein theoretisch müßte man alle möglichen vorkommenden Bitmuster in ein RAM einschreiben, um es vollständig zu testen. Das ist aber praktisch unmöglich. Beispiel: Wir haben z. B. nur 1024 Speicherzellen. Es gibt dann 2^{1024} verschiedene mögliche Datenmuster in diesem Speicher. Wenn man für ein Bit 1 µs benötigt, um es einzuschreiben, so benötigt man für den ganzen Speicher für ein mögliches Bitmuster
$1\ \mu s \times 1024 = 1{,}024\ ms$
Das geht noch recht gut.
Nun müssen wir aber alle Kombinationsmöglichkeiten einschreiben, also:
Gesamtzeit $= 1{,}024\ ms \times 2^{1024}$
In *Bild 10* ist die Zahl 2^{1024} dargestellt. Sie lautet abgekürzt: $1{,}79769 \times 10^{312}$.
Nun multiplizieren wir das Ergebnis mit 1,024 ms und erhalten
$1{,}84083 \times 10^{312}$ ms.
Wir wollen das Ergebnis einmal in Jahren ausrechnen. Dazu wird es erst durch 1000, dann durch 3600 und dann durch 365 geteilt. Es ergibt sich $1{,}4009 \times 10^{303}$. Eine Zahl mit 303 Stellen. Die Zahl der Jahre ist unvorstellbar groß, das Universum besitzt eine wesentlich geringere Lebenserwartung. Auch wenn der Rechner zehn- oder hundertmal so schnell die Daten in das RAM einschreiben würde, könnte man nicht alle Kombinationen prüfen, denn an dieser riesigen Zahl ist kaum etwas zu ändern.

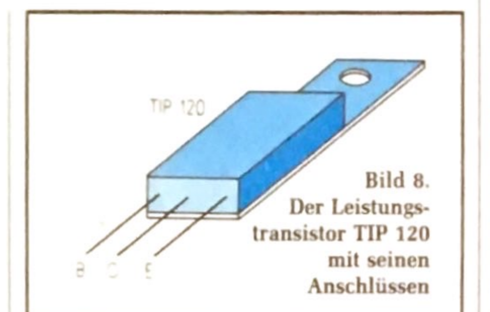


Bild 8. Der Leistungstransistor TIP 120 mit seinen Anschlüssen

Das bedeutet übrigens auch, daß niemand jemals alle möglichen Programme in RAM-Bausteine einbringen könnte. Dabei wurde ein Speicher mit nur 1024 Bit zugrunde gelegt. In der SBC2 gibt es dagegen zwei Bausteine mit zusammen 32 768 Bits, die Vielfalt ist also dort ungeheuer groß.
Übrigens ist die Zeit zum Auslesen eines Bitmusters, denn zum Auslesen eines Bitmusters benötigt man natürlich auch eine bestimmte Zeit, noch nicht mitgerechnet. Die große Zahl muß also noch verdoppelt werden.

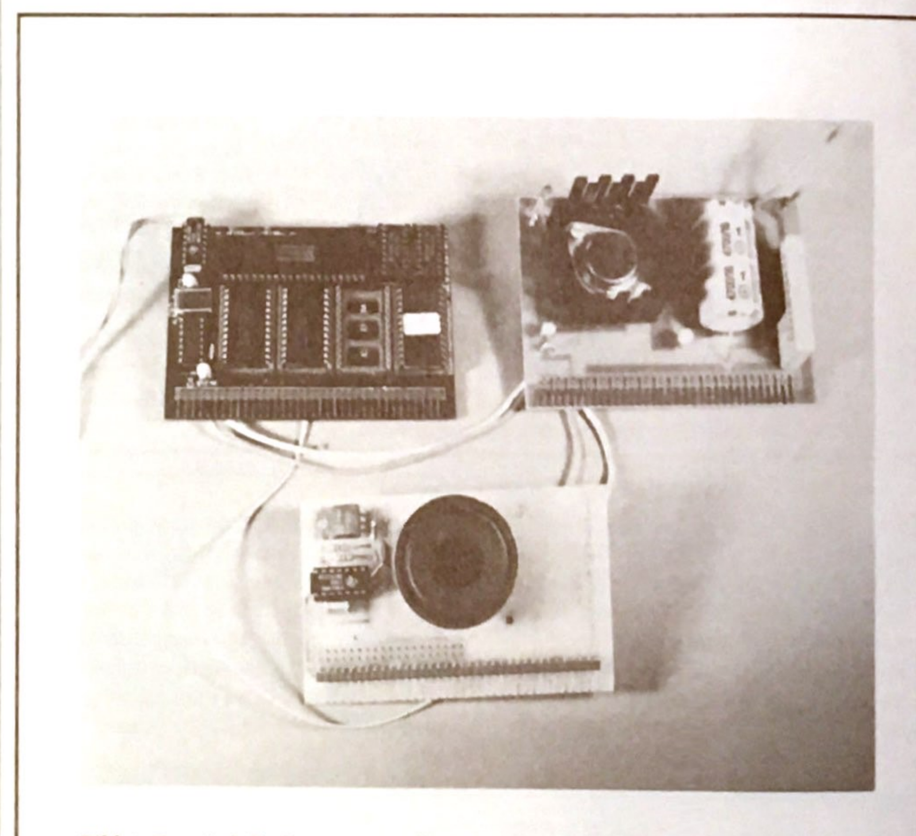


Bild 9. Das sind die Baugruppen, die mitspielen: POW5V, SBC2 und „Musik". Die Verbindungen und die Stromversorgung laufen über Schaltdraht

179769313486231590772930519078902473361797697894230657273430081157732675805500 9
6313270847732240753602112011387987139335765878976881441662249284743063947412437
7767893424865485276302219601246094119453082952085005768838150682342462881473913
110540827237163350510684586298239947245938479716304835356329624224137216

Bild 10. Die Zahl 2^{1024} dezimal dargestellt

Beim Testen von RAMs kann man also nur eine verschwindende Anzahl von Kombinationen testen. Zugute kommt, daß man den Aufbau der RAM-Zellen kennt und daß Kurzschlüsse zum Beispiel nur zwischen Nachbarzellen vorkommen und nicht zwischen weit entfernten Speicherzellen, so daß man heute RAM-Bausteine schon gut testen kann.
Vor einigen Jahren war es noch an der Tagesordnung, daß man defekte RAMs von Herstellern geliefert bekam, die irgendeinen seltsamen Fehler aufwiesen. Heute sind die RAMs in der Qualität sehr gut, man muß kaum noch mit Fehlern rechnen.

Oszilloskop
Externe-Triggerung
Trigger-Eingang
Y-Eingang
Masse
Prüfspitze
RD, WR, Decoder, Daten, IORQ
SBC II

Bild 11. Die Meßanordnung. Am besten geht es mit einem Zweistrahl-Oszilloskop

Ein „einfacher" Test

Wir wollen einen einfachen Test einmal mit unserer SBC2-Karte durchführen. Dazu gibt es das EPROM RWT, das man im Handel beziehen kann. Das Programm ist auch im Buch „Mikroprozessor selbstgebaut und programmiert" abgedruckt (unter Abbildung 7.1.6). Das EPROM wird mit der Markierung zur Steckerleiste hinzeigend in die Fassung 0 (IC 6, anstelle des MUO-EPROMs) eingesteckt.
Dann wird der Rechner eingeschaltet. Nun kommt ein etwas komplizierter Meßvorgang, der nur von Oszilloskop-Besitzern durchgeführt werden kann. *Bild 11* zeigt die Meßanordnung. Der Triggereingang des Oszilloskopes wird an das $\overline{IORQ}$-Signal geführt. Dort erscheint per Programm ein kurzer Puls, um den Meßvorgang zu starten. Die Triggerung muß dabei auf DC, Extern umgeschaltet sein.

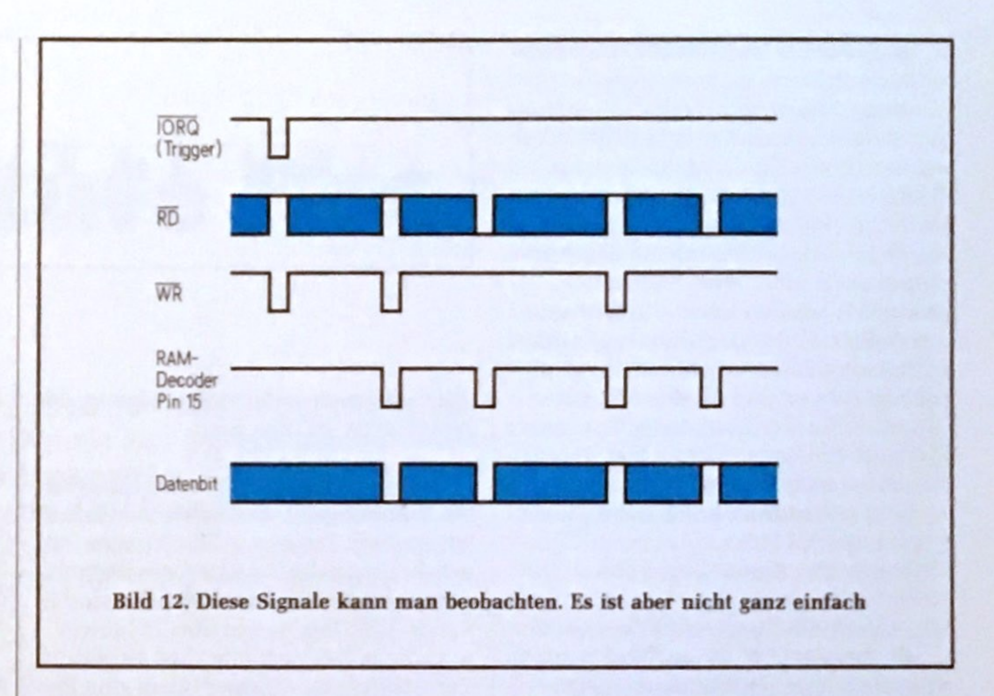


Bild 12. Diese Signale kann man beobachten. Es ist aber nicht ganz einfach

Mit dem Y-Eingang führt man die Messung durch. Nur mit einem Zweikanal-Gerät ist das Messen hier bequem möglich. Die Zeitbasis stellt man auf 2 µs/cm und den Abschwächer auf 2 V/cm.
Auf dem Schirm zeigen sich die einzelnen Signale nach *Bild 12*. Die dunklen Flächen bei Signalen sollen bedeuten, daß dort der Wert des Signales beliebig sein darf. Dort sind sehr viele unterschiedliche Pulse untergebracht, die für den Test keine Bedeutung haben.
Am RAM-Decoder (74LS138, IC 5) Pin 15 erscheint das Aktivierungssignal für den RAM-Baustein in Sockel 2. Nur wenn dieser Puls auf 0 V liegt, sind die restlichen Signale zu beachten. Man kann einmal den $\overline{WR}$-Ausgang der CPU (Pin 22 der CPU) ansehen und das $\overline{RD}$-Signal (Pin 21 der CPU). Wir sehen, daß zuerst geschrieben wird, dann gelesen, dann wieder geschrieben und dann wieder gelesen.

Nun kann man sich den Datenbus (z. B. Pin 9 des RAM-Speichers) zu diesen Zeiten ansehen.
Beim Schreiben liegt zuerst ein 0-Signal an, beim Lesen erscheint dieses dann wieder auf dem Datenbus, dann wird eine 1 eingeschrieben, die danach wieder gelesen wird. Es ist nicht ganz einfach, diese Signale zu messen, da viele verwirrende Pulse auf den Datenleitungen vorhanden sind. Denn die CPU benutzt die Datenleitungen für verschiedene Aufgaben. Man nennt diese Datenleitungen auch Datenbus, tatsächlich vom lateinischen Wort Omnibus (für alle) abgeleitet, da der Datenbus von vielen Bausteinen gemeinsam benutzt wird.

Bus: Leitungspaket für viele Bausteine

Um mit diesem Bus arbeiten zu können, benötigen die angeschlossenen Baustei-

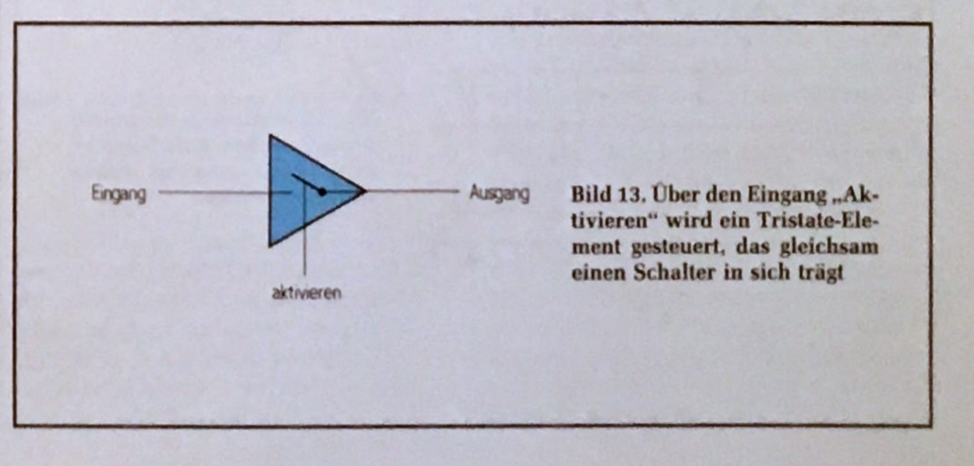


Bild 13. Über den Eingang „Aktivieren" wird ein Tristate-Element gesteuert, das gleichsam einen Schalter in sich trägt

ne ein spezielles Bauelement, das Tristate-Element. Wenn mehrere logische Ausgänge integrierter Bausteine einfach so zusammengeschaltet sein würden, so ergäbe sich ein Kurzschluß, wenn ein Element 1 ausgibt, während das andere 0 führt. Ein Tristate-Element ist nun ein Baustein, dessen Ausgang mit den Ausgängen anderer Tristate-Elemente verbunden werden kann. Ein Auswahlsignal (bei uns der Decoder) sorgt dafür, daß immer nur ein Tristate-Element zur Zeit aktiviert ist und 0- oder 1-Pegel ausgibt, während die anderen Tristate-Elemente sich so verhalten, als seien sie überhaupt nicht vorhanden. Sie haben weder den Zustand 0 am Ausgang, noch 1, sondern sind Tristate, wie man sagt. Sie haben also einen dritten Zustand.

Man kann sich das so vorstellen, als ob solche Bauelemente einen Schalter am Ausgang hätten, mit dem man sie bei Bedarf an den Bus anschalten und davon trennen kann. *Bild 13* zeigt ein Schema. Mit dem Eingang „aktivieren" wird der Schalter geschlossen. Dann gelangt das am Eingang stehende Signal direkt an den Ausgang. Ist der Schalter offen, so ist auch der Ausgang abgetrennt und überläßt einem anderen Bauelement, das gerade aktiv ist, den Bus.



Bild 16. So sehen die Impulse aus, die einen mehrstimmigen Klang herbeizaubern. Das EPROM MUM erzeugt sie

Natürlich befinden sich in den integrierten Schaltungen keine wirklichen Schalter, sondern Transistor-Schaltungen, mit welchen man das Tristate-Verhalten hervorrufen kann. Tristate-Elemente sind in RAMs, EPROMs und in die CPU sowie in anderen Bauelementen fest eingebaut. Der „Aktivieren-Eingang" kann zum Beispiel über einen Decoder gesteuert werden und die CPU teilt dann durch eine Adresse mit, welcher Baustein aktiviert werden soll. Ihre eigenen Tristate-Elemente kann die CPU selbst ein- und ausschalten. In der Regel bestimmt sie, wer den Datenbus gerade benutzen darf.

Beethoven aus dem Computer

Genug der Theorie, jetzt sei wieder ein EPROM ausprobiert, das EPROM „MUM". Es wird ein Menuett von Ludwig van Beethoven mehrstimmig gespielt. Das EPROM MUM funktioniert nur, wenn auch RAM-Bausteine vorhanden sind.

Wie kann man aber auf unserem System mehrstimmig spielen, wenn man nur einen $\overline{\text{IORQ}}$-Ausgang hat? Das Geheimnis zeigt sich in *Bild 14*. Dort sind einmal vier Stimmen übereinander gezeichnet. Nun kann man die Schwingungen stark verändern, wie es *Bild 15* zeigt. Dort wird anstelle der symmetrischen Schwingung nur jeweils ein kurzer Puls gegeben. Dadurch ist genügend Platz zwischen den Pulsen, so daß man alle Pulse einfach „übereinanderlegen" kann. Man erhält *Bild 16*.

Daß man aus diesem Gemisch die Musik wieder heraushören kann, ist ein wahres Wunder, doch es geht. Da hier die Pulse sehr schmal sind und pro Schwingung nur noch ein Puls auftaucht, benötigt man wieder das Monoflop. Erst dann ergibt sich ein passables Ergebnis. Die Lautstärke ist aber doch geringer als bei den Impulspaketen aus dem ersten Versuch. Übrigens kann man beide Musikstücke selbst verändern, wenn man die Notentabellen im EPROM ändert. Dazu sollte man jedoch jemanden heranziehen, der sich mit der Programmierung von EPROMs schon auskennt, denn mit Ihrem jetzigen Wissensstand ist das leider noch nicht möglich.

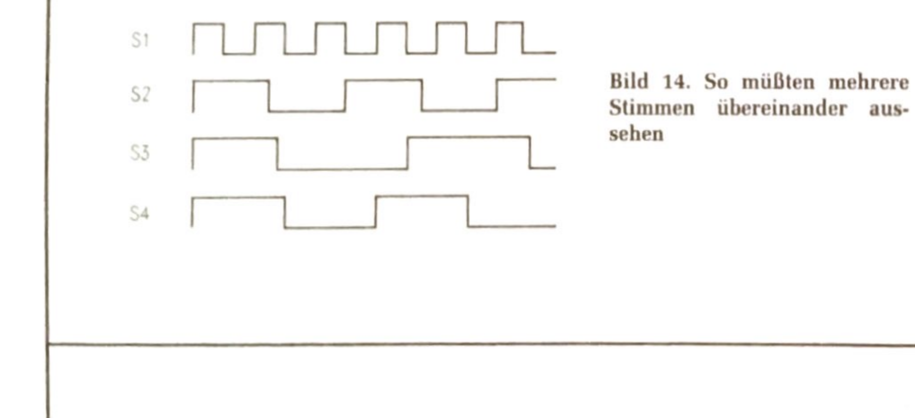


Bild 14. So müßten mehrere Stimmen übereinander aussehen

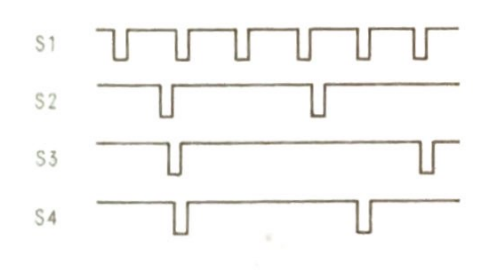


Bild 15. Wenn man die Impulse stark unsymmetrisch macht, dann kann man sie später übereinanderlegen

Aufgaben

1. Was ist ein Tristate-Element? Wozu braucht man es?

2. Was bedeutet das Wort Bus in dem Begriff Datenbus?

Rolf-Dieter Klein

Alarmstufe Rot

Mikroelektronik, Folge 9

In der letzten Folge konnte der Z80 zeigen, daß er gezielt Signalwechsel erzeugen kann. Diese Signale aus seinem Inneren wurden mit wenigen zusätzlichen Bauelementen hörbar gemacht. In diesem Kapitel wird nun das, was der Mikrocomputer ausrechnet, gezielt nach außen gegeben, in einen eigenen Baustein gespeichert und als Schaltsignal für eine Ampelanlage verwendet.

Immer mehr Ampelanlagen in Großstädten werden von Computern gesteuert. Der Vorteil ist dabei, daß mehrere Ampeln von einem zentralen Rechner je nach Verkehrslage optimal geschaltet werden können. Voraussetzung dafür ist, daß die Steuerung der einzelnen Ampelanlagen digital durchgeführt wird. Denn dadurch kann man ohne viel Aufwand, allein durch Ändern eines Programmes, den Schaltrhythmus der Anlage allen Erfordernissen anpassen.

Das Ampel-EPROM

Wir wollen eine solche Ampelsteuerung mit der SBC2-Baugruppe aufbauen. Allerdings muß der Computer dafür erweitert werden.
Wir benötigen natürlich ein Steuerprogramm, das in einem EPROM enthalten ist, das bereits fertig programmiert ist. Das EPROM mit der Ampelsteuerung trägt die Aufschrift AST. Es ist im Handel erhältlich.
Das Ampel-EPROM (AST) wird in den Sockel 0 (IC6) der SBC2-Baugruppe eingesetzt. Ein RAM-Baustein (6116) bleibt in Sockel 2 (IC8). Wir schalten den Computer einmal probeweise ein und messen die Impulse.
An Pin 20 der Z80-CPU ist der $\overline{\text{IORQ}}$-Ausgang. Dort erscheinen kurze Impulse im Abstand von etwa 2 s. Wir messen den Impuls am besten mit dem Prüfstift. Die Leuchtdioden W1 und W2 wechseln sich bei jedem Puls ab. Die Leuchtdiode H brennt praktisch dauernd. Die Leuchtdiode L zeigt zwar den kurzen Puls an Pin 20 an, bleibt dabei aber für das Auge so dunkel, daß man den Puls nicht sehen kann. Wenn W1 und W2 abwechseln blinken, dann arbeitet unser Ampel-EPROM bereits. Zu sehen sind die Ampelschaltphasen.

Die Ampelschaltung

Die Ampel selbst soll aus vier Gruppen von roten, gelben und grünen Leuchtdioden bestehen. *Bild 1* zeigt die Anordnung der Ampeln an einer Kreuzung. Die jeweils gegenüberstehenden Ampeln zeigen gleiche Ampelphasen und werden daher gleich beschaltet. Wir benötigen also nur zwei Gruppen von Treibern, also insgesamt 6 Treiber, zwei für rot, zwei für gelb und zwei für grün. *Bild 2* zeigt die komplette Schaltung. Dort ist noch ein bisher unbekannter Baustein eingezeichnet, der gleich geschildert werden soll.

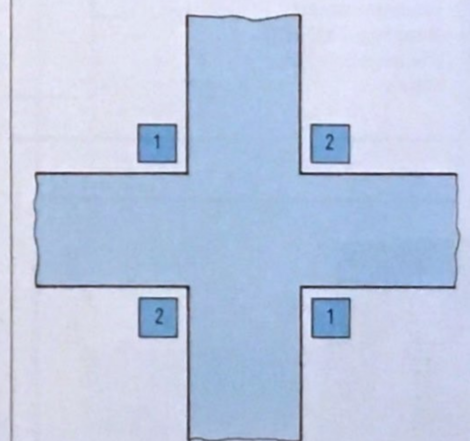


Bild 1. Das Schema einer Ampelanlage

Wir haben nämlich ein Problem. Wir haben bisher nur einen einzigen Schaltimpuls als Ausgabe-Signal gemessen, das $\overline{\text{IORQ}}$-Signal, müssen jetzt aber 6 Treiberstufen gleichzeitig ansteuern. Wie kann man das erreichen? Der Z80 besitzt einen sogenannten Datenbus. Das sind die Leitungen D0, D1, D2, D3, D4, D5, D6 und D7. Über diesen Datenbus kann der Z80 sowohl Informationen einholen als auch ausgeben. Wenn man sich den Datenbus beim Z80, genauer gesagt, einen Pin desselben, einmal mit dem Oszilloskop ansieht (*Bild 3*), so erkennt man, daß sich darauf eine Vielzahl von Signalen findet. Das rührt daher, daß alle wichtigen Informationen für den Z80 über diese acht Leitungen laufen: Die Befehle, die Daten, die auszugebenden Signale und das, was eingegeben werden soll. Wann die Daten für die Ausgabe bestimmt sind, das regelt ein Zusatzsignal, das schon im vorherigen Kapitel, mit etwas artfremder Bestimmung, verwendet wurde: das $\overline{\text{IORQ}}$-Signal. Immer wenn das $\overline{\text{IORQ}}$-Signal auf 0 V liegt, das geschieht für sehr kurze Zeit, legt der Z80 auf den Datenbus eine Information, die mit der Außenwelt zu tun hat – oder er erwartet eine solche. Eigentlich gehören noch ein zweites und ein drittes Signal dazu, die wir auch schon kennen, nämlich das $\overline{\text{WR}}$-Signal, und das $\overline{\text{RD}}$-Signal. $\overline{\text{WR}}$ bedeutet, die CPU will Daten schreiben, also vom Inneren der CPU nach außen geben. Und das $\overline{\text{WR}}$-Signal erscheint auch, wenn man mit dem Prüfstift bei laufendem Ampelprogramm an Pin 22 mißt. Es ist dort dann am Wechsel von W1 und W2 zu sehen.

Was ein Latch ist

Die Information für die Außenwelt liegt also am Z80 immer nur sehr kurz, jeweils für die Zeitdauer der $\overline{\text{IORQ}}$-Pulse, auf dem Datenbus. Der Baustein 74LS374 ist dafür konstruiert, solche Informationen aufzunehmen und zwischenzuspeichern. Man nennt ihn daher auch Zwischenspeicher. In Neudeutsch nennt man das Latch. In dem Baustein befinden sich 8 einzelne Zwischenspeicher-Elemente, sogenannte D-Flip-Flops. Immer wenn an Pin 11 des Bausteines, dem Takteingang, ein Signalwechsel von 0 auf 1 stattfindet, werden die Informationen, die an den Eingängen D0 bis D7 liegen, übernommen und solange gespeichert, bis ein neues Byte, gesteuert durch den Takteingang, übernommen wird. An den Q-Ausgängen liegt die Information also dauerhaft von

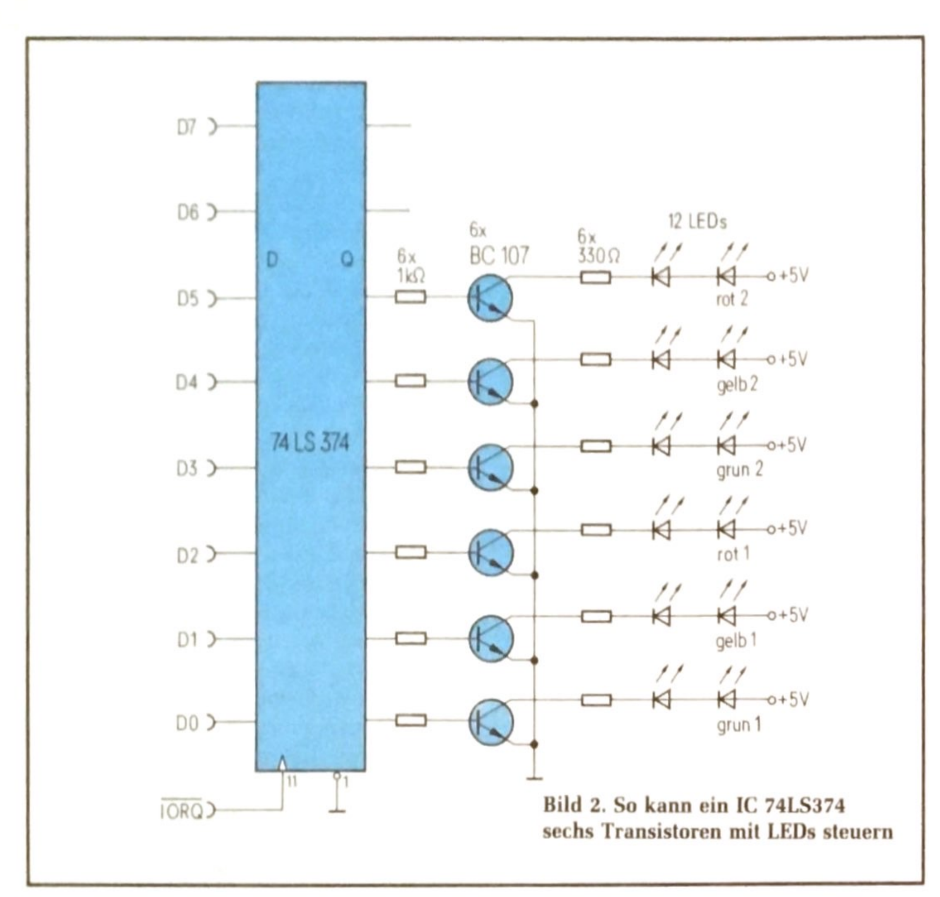


Bild 2. So kann ein IC 74LS374 sechs Transistoren mit LEDs steuern

Signalwechsel zu Signalwechsel an. *Bild 4* zeigt ein Beispiel. Der Q-Ausgang beginnt mit ???-Werten, weil man nicht weiß, was vor Beginn des Betrachtungszeitraumes eingespeichert war. Nach dem ersten sichtbaren Signalwechsel von 0 auf 1 an Pin 11 ist aber klar, was am Ausgang liegt: Genau das an den D-Eingängen anstehende Signal.
Der Baustein 74LS734 besitzt noch einen weiteren Eingang, nämlich Pin 1.

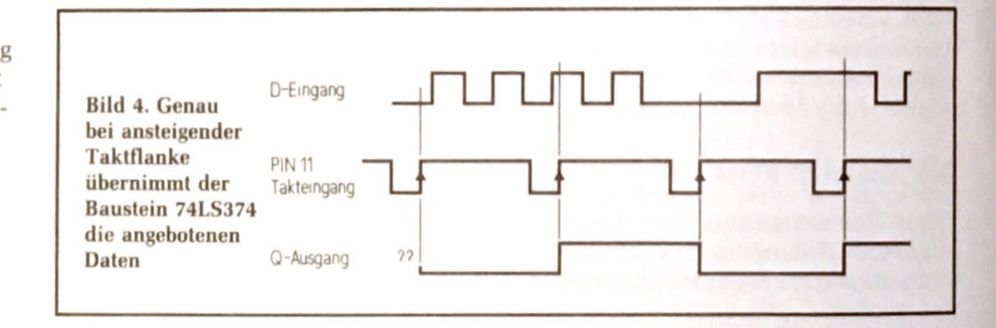


Bild 4. Genau bei ansteigender Taktflanke übernimmt der Baustein 74LS374 die angebotenen Daten

Nur wenn dieser Pin auf 0-Signal liegt, erscheinen am Ausgang Informationen, sonst nimmt der Ausgang Tristate-Zustand an. Er bleibt offen. Man hat die Tristate-Elemente an den Ausgängen des 74LS374 mit eingebaut, um den Baustein universell verwenden zu können. Zum Beispiel auch, um über den Datenbus Signale an einen Mikroprozessor abgeben zu können.

Von Zusatzkarten und vom System-Bus

Im Prinzip ist jetzt alles klar. Der Z80 schreibt die Stellung der Ampelanlage in den Zwischenspeicher, der dann Verstärker bedient, die die einzelnen Lampen ein- und ausschalten.
Zu unserem Mikrocomputersystem gibt es nun eigene Baugruppen, die den Anbau von externen Elementen erleichtern. Als erstes sei die Karte IOE genannt, die unter anderem einen Baustein 74LS374 enthält, den wir hier benutzen (*Bild 5*). Außerdem gibt es eine sehr wichtige Karte, die Grundplatte, die System-Bus-Platte genannt wird, weil sie Leitungen des Systems bereithält und diese an mehrere Steckplätze heranführt (*Bilder 6, 7*). Es gibt auch einen eigenen Bausatz zur Ampel-Steuerung, der mit IOE zusammenpaßt.
Mit der System-Bus-Karte kann man nun alle anderen Karten miteinander verbinden. Bild 6 zeigt die Unterseite der Bus-Karte. Man sieht, daß fast alle Leitungen einfach parallel durchgehen. Sie verbinden die Stifte der Baugruppen parallel miteinander.
Auf der Oberseite der Busplatte, die *Bild 8* zeigt, sind Buchsenleisten angebracht. Beim Einlöten dieser Buchsenleisten sollte man darauf achten, daß man sie flach anliegend einlötet. Am Besten geschieht das, indem man zunächst zwei Ecken anlötet, dann die Oberseite betrachtet und ggf. die Lötungen korrigiert.

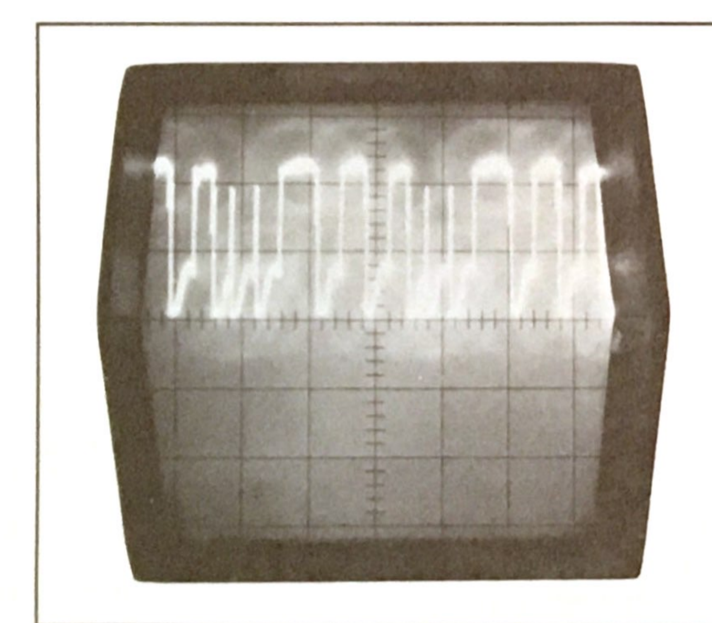


Bild 3. Das sind Signale, wie sie sich auf dem Bus herumtreiben

Nun kann man noch Halterungen anbringen, die den Baugruppen Führung geben sollen. Auf die Orientierung der Karten, die eingesteckt werden, muß geachtet werden. Dazu erkennt man auf der Unterseite der Busplatte zwei Leiterbahnen, die dicker sind als alle anderen. Auf den Baugruppen finden sich ebenfalls zwei solche Anschlüsse. Beim Einstecken müssen diese Kontakte miteinander verbunden werden, denn es handelt sich um die +5-V-Leitung und den Masseanschluß.

Wenn man die POW5V-Baugruppe verkehrt einsteckt, können andere Baugruppen zerstört werden, daher unbedingt die Anschlüsse kontrollieren. Bild 6 zeigt, wie es aussehen soll, wenn mehrere Baugruppen eingesteckt sind.
Wo die einzelnen Karten eingesteckt werden, also auf welcher Buchsenreihe, spielt vorerst keine Rolle. Auch bleiben einige Buchsen noch frei, da sie erst für spätere Ausbaustufen benötigt werden.

Die IOE-Karte

Grundsätzlich ist es so, daß viele Mikroelektronikschaltungen erschütternd einfach aufzubauen sind. Ein bißchen Lochraster-Platine, ein bißchen lötbarer Kupferlackdraht, die Bauelemente und ein bißchen Mut und Geduld und schon steht die Schaltung. Allerdings benötigt man auch etwas Übersicht dabei, denn es sind doch sehr viele Verbindungen zu schlagen, so daß der eine oder andere durchaus nervös werden kann, wenn er nach Schaltplan von Pin zu Pin die Drähte stricken soll. Einfacher ist es da schon, eine vorgefertigte Platine zu kaufen, auf der alle Lötpunkte vorgegeben sind und die Lage der Bauelemente meist eindeutig festgelegt ist. Deshalb gibt es viele Zusatzkarten zum NDR-Klein-Computer.
Im System der Karten gibt es wie gesagt die IOE-Karte, die es erlaubt Signale aus dem Computersystem nach außen zu schleusen und ebenfalls von außen Signale an den Computer heranzutragen. Sie enthält zum Beispiel (unter anderem) zwei Bausteine des vorhin besprochenen ICs 74LS374. Außerdem enthält sie noch einige zusätzliche Bauelemente, die erst später im Kurs eine Rolle spielen werden. Der Gesamtschaltplan, den *Bild 9* zeigt, ist vielleicht noch nicht ganz einfach zu verstehen. Zunächst muß man wissen, daß der Z80 nicht nur dann eine Adresse auf seine Adreßleitungen aussendet, wenn er den Speicher ansprechen will, sondern auch, wenn er das $\overline{\text{IORQ}}$-Signal bedient.

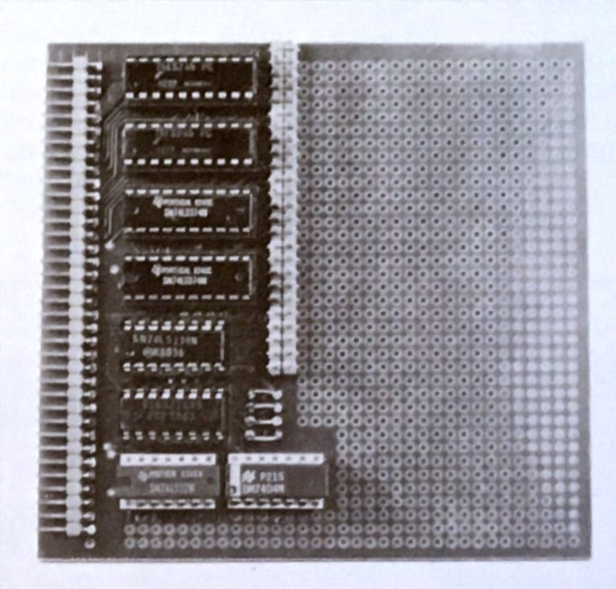
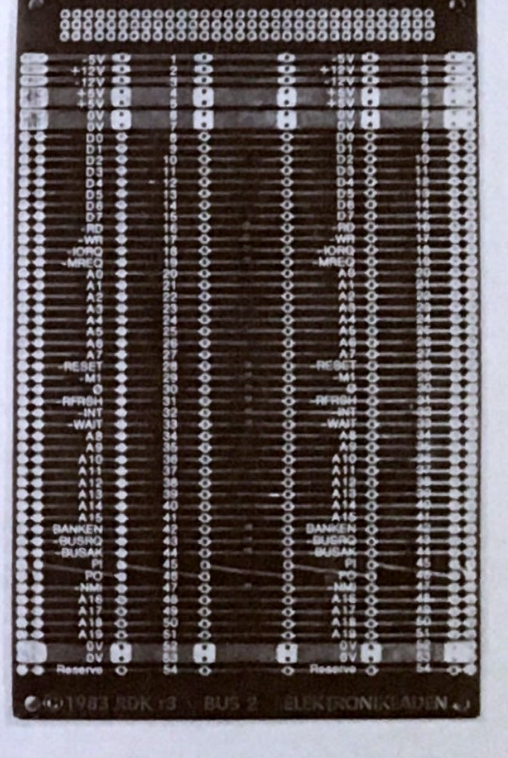


Bild 5. Die Platine für Ein- und Ausgabe von Daten



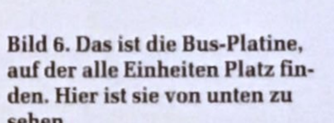


Bild 6. Das ist die Bus-Platine, auf der alle Einheiten Platz finden. Hier ist sie von unten zu sehen

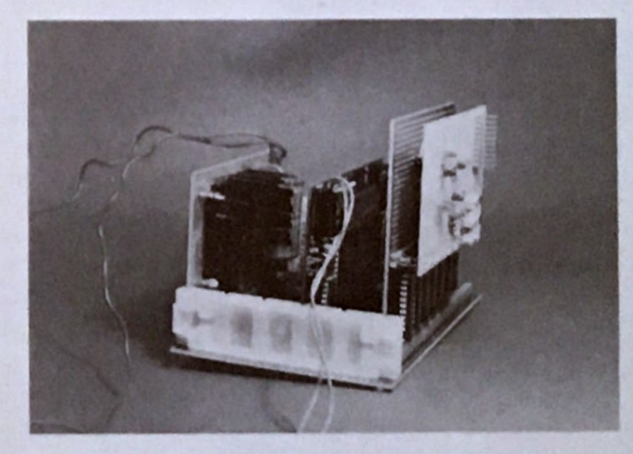


Bild 7. Die Busplatine mit Netzteil, SBC2 und IOE mit Ampelanlage

Das hat den Vorteil, daß er mehrere verschiedene Geräte ansprechen kann, wenn diese wieder geeignete Decoder eingebaut haben. Das Schaltbild zeigt, daß auf die IOE-Karte sowohl die acht Datenleitungen, als auch sechs der insgesamt 16 Adreßleitungen an die Busplatine angeschlossen sind.

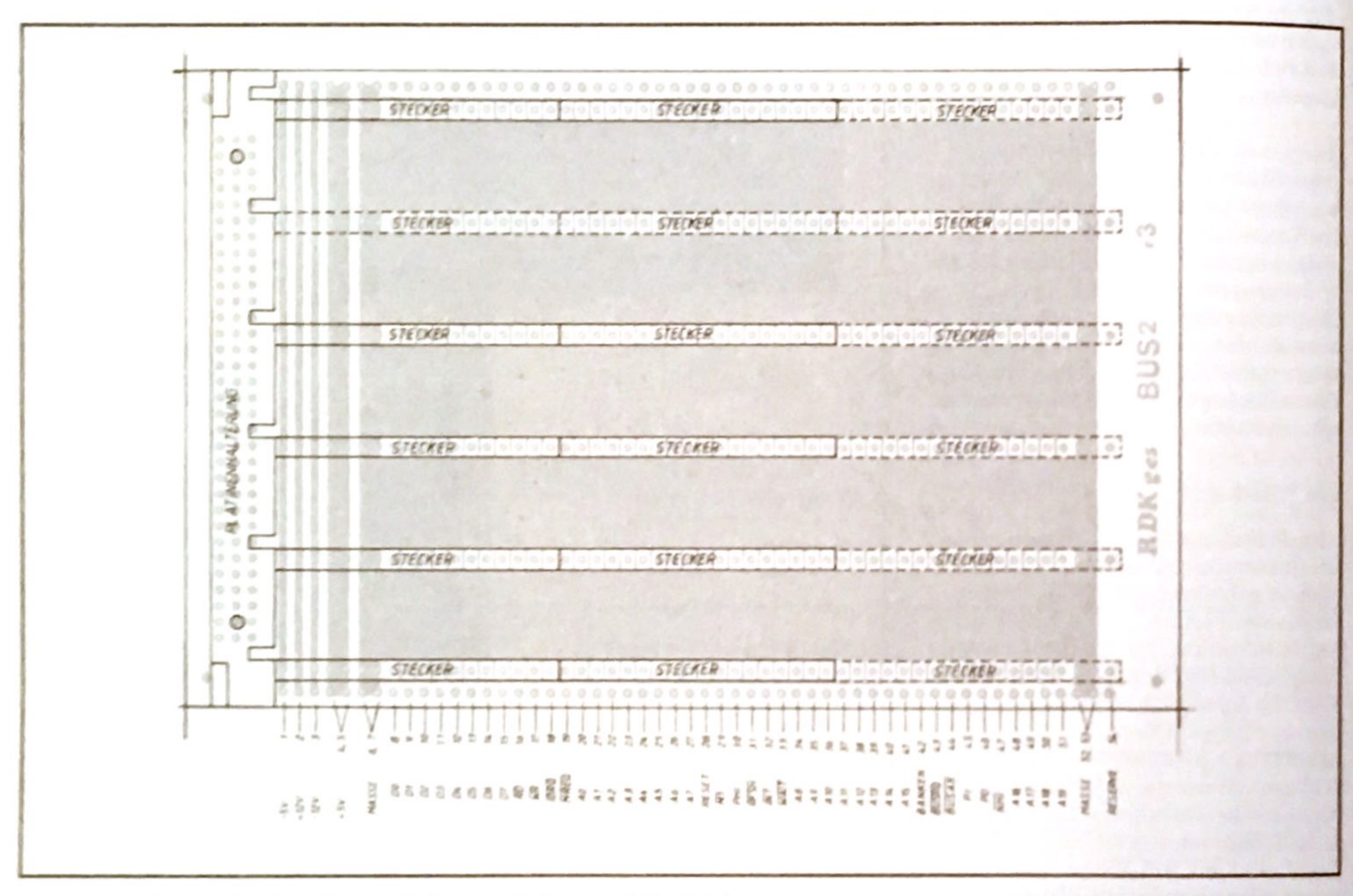


Bild 8. Die Oberseite der Bus-Platine und die Signalbelegung auf den Steckern

Betrachten Sie zunächst den unteren Teil links der Schaltung. Da wird das $\overline{IORQ}$-Signal so herangeführt (über einen Inverter), daß es den Baustein 7 genau dann aktiviert, wenn $\overline{IORQ}$ selbst aktiv ist. Baustein 7 ist ein IC namens 74LS85. Seine Funktion ist ganz schön raffiniert: es besitzt die Fähigkeit, die Signale, die an den Pins 10, 12, 13, 15 anliegen mit den Signalen zu vergleichen, die an die Pins 9, 11, 14, 1 angelegt werden. An seinem Pin 6 gibt es genau dann eine 1 aus, wenn das Bitmuster, das von den vier Adreßbits links angeboten wird, mit dem Bitmuster, das mit den Lötbrücken rechts im „Jumper-Feld-1" eingestellt werden kann, übereinstimmt und wenn der Eingang IN aktiv ist. Sie selbst können durch Einlöten von Drahtbrücken, die den entsprechenden Eingang des „Vergleichers" auf 0 ziehen, bestimmen, bei welcher Adresse er sich melden soll, beziehungsweise den Rest der Schaltung aktivieren soll. Offene Eingänge dieses ICs wirken dabei so, als sei daran 1 angelegt. Wenn Sie für die Ampelsteuerung vorerst alle vier Eingänge hier auf 0 legen, können Sie später durch Auftrennen immer noch eine beliebige andere Adresse einstellen.

Über den Inverter 74LS04 (da sind mehrere in einem IC) wird das Ausgangssignal des Vergleichers so präpariert, daß es über die folgenden ODER-Bausteine im IC 74LS32 geeignet mit $\overline{WR}$ und $\overline{RD}$ verknüpft werden kann. An dieser Stelle kommt eine kleine Schwierigkeit: Ein ODER führt an seinem Ausgang genau dann eine 1, wenn einer der Eingänge eine 1 führt. Wenn man das anders ausspricht: Ein ODER führt an seinem Ausgang genau dann eine Null, wenn alle Eingänge auf 0 liegen. Im gewissen Sinn ist ein ODER also mit dem UND verwandt. Hier wird diese Verwandschaft dazu ausgenutzt, die nächsten Bausteine freizugeben. Wenn nämlich der $\overline{WR}$-Impuls aktiv ist, liegt er auf 0 und ein Freigabe-Signal aus dem Vergleicher bewirkt, daß die rechte Hälfte des 74LS32 auf beiden Eingängen 0 bekommt, was zu einer 0 an seinem Ausgang, der vorher auf 1 lag, führt. Das wiederum schaltet den Baustein 74LS139, „rechte Hälfte", frei. Dieser Schaltungsteil enthält einen Decoder mit nur zwei Eingängen A, B, die insgesamt vier Ausgänge aktivieren können.

Wenn also sowohl das $\overline{IORQ}$-Signal aktiv ist, als auch die mit den Drahtbrükken in JMP eingestellte Adresse mit der vom Prozessor auf A4, A5, A6, A7 eingestellten übereinstimmt und das $\overline{WR}$-Signal aktiv ist, dann wird eins der beiden Signale W0 oder W1 aktiv. Welches von beiden, das hängt von den Bits A0 und A1 ab, die der Prozessor liefert.

Ist alles wie oben im Ablauf, nur daß das $\overline{RD}$-Signal aktiv, dann wird die „linke Hälfte" der beiden ICs angesprochen und eine der Leitungen R0 und R1 aktiv. Ein Impuls auf W0 zum Beispiel bewirkt, daß IC 4 die Daten vom Datenbus her übernimmt. Ein Impuls auf W1 veranlaßt IC 5 zur Datenübernahme. Und

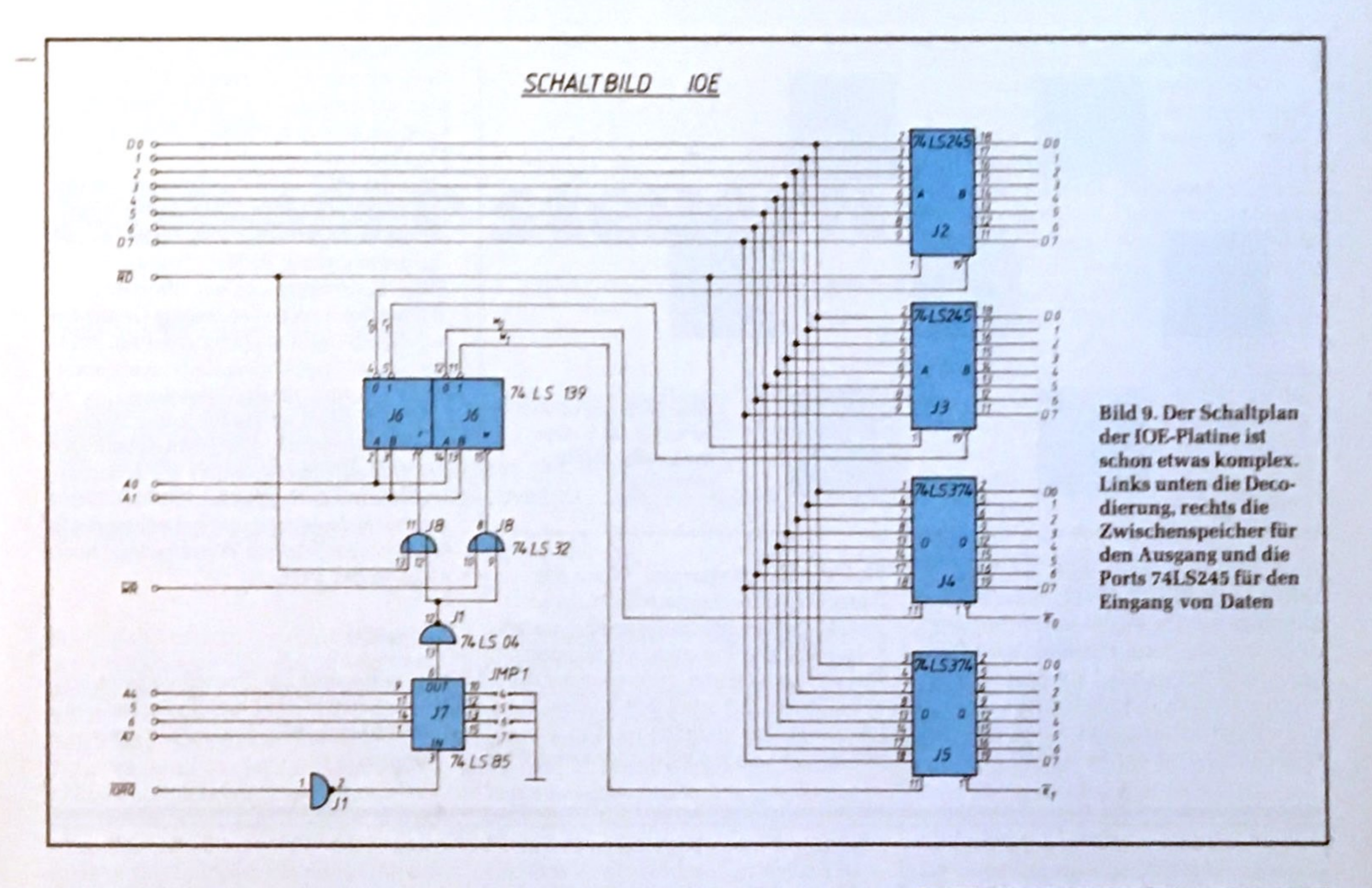


Bild 9. Der Schaltplan der IOE-Platine ist schon etwas komplex. Links unten die Decodierung, rechts die Zwischenspeicher für den Ausgang und die Ports 74LS245 für den Eingang von Daten

genau diese Latches benötigen wir für die Ampelsteuerung. Die anderen ICs, die 74LS245 werden durch $\overline{RD}$ freigeschaltet. Was sie tun soll, wird erst später erklärt werden.

Es ist noch zu sagen, daß die Adressenleitungen A2 und A3 zunächst nicht mit der Schaltung ausgewertet werden. Das bedeutet, daß die Schaltung sich bei mehreren Adressen meldet.

Die Ampelschaltung wird aufgebaut

Im Grunde könnten Sie jetzt die Ampelschaltung auf dem Lochrasterfeld der IOE-Platine aufbauen. Wer das tun will, muß sich mit der Belegung der zwei Lochreihen beschäftigen, an die die Aus- bzw. Eingänge der IOE-Platine zur Weiterverwertung durch den Benutzer herangeführt werden. *Bild 10* zeigt die Bitnummern die an den Kontakten liegen. Vielleicht ist es auch lustig für Sie, das Ampelprom laufen zu lassen und nur mit einem Vielfach-Meßinstrument oder mit dem Teststift an den Bits der IOE-Platine zu verfolgen, was geschieht.

Zum Test

Wenn keine Ampelphasen ausgegeben werden, dann betrachtet man das $\overline{IORQ}$-Signal. Es muß an Pin 11 des 74LS374 ankommen. W1 und W2 müssen im Sekundenabstand wechselweise blinken. An den Q-Ausgängen (gegenüber D0 bis D5) müssen die Ampelsignale sichtbar sein, die LEDs L und H müssen da die Ampelphasen anzeigen. Dann kann man noch die isolierten Treiber testen, indem man die Verbindung zwischen Treiber und IC auftrennt (z. B. IC herausziehen) und +5 V an die Treibereingänge legt, die LEDs müssen dann jeweils leuchten. Wenn Sie noch nicht sehr geübt im Löten und verdrahten sind, dann sollten Sie die Ampelanlage unbedingt aufbauen. Sie ist schön einfach, billig und als Übungsobjekt sehr geeignet.

Nun fehlt eigentlich nur noch die Koppelung von mehreren Ampelsteuerungen zu einer Anlage mit mehreren Kreuzungen.

Grüne Welle

Sie können hierzu einen Versuch durchführen, wenn Sie in einer Arbeitsgruppe zwei und mehrere Ampeln aufgebaut haben. Einer aus der Gruppe stellt dann seinen Computer als Steuerrechner für die anderen Computer zur Verfügung. *Bild 11* zeigt die Verschaltung. In den

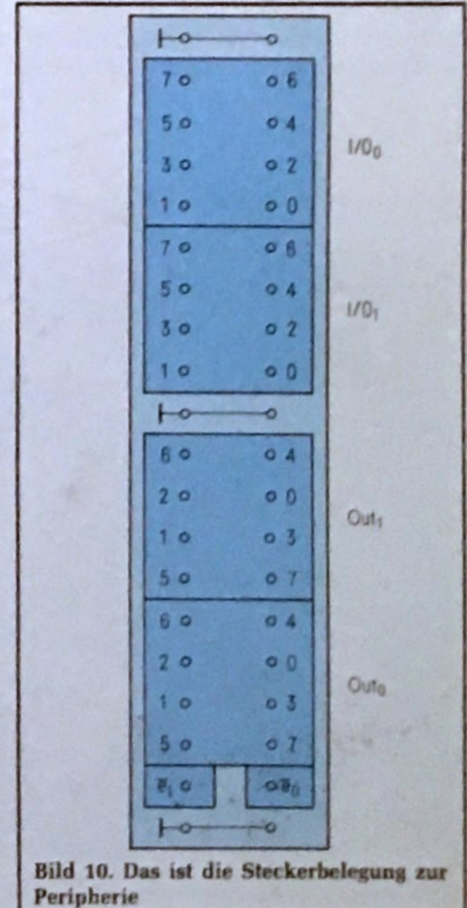


Bild 10. Das ist die Steckerbelegung zur Peripherie

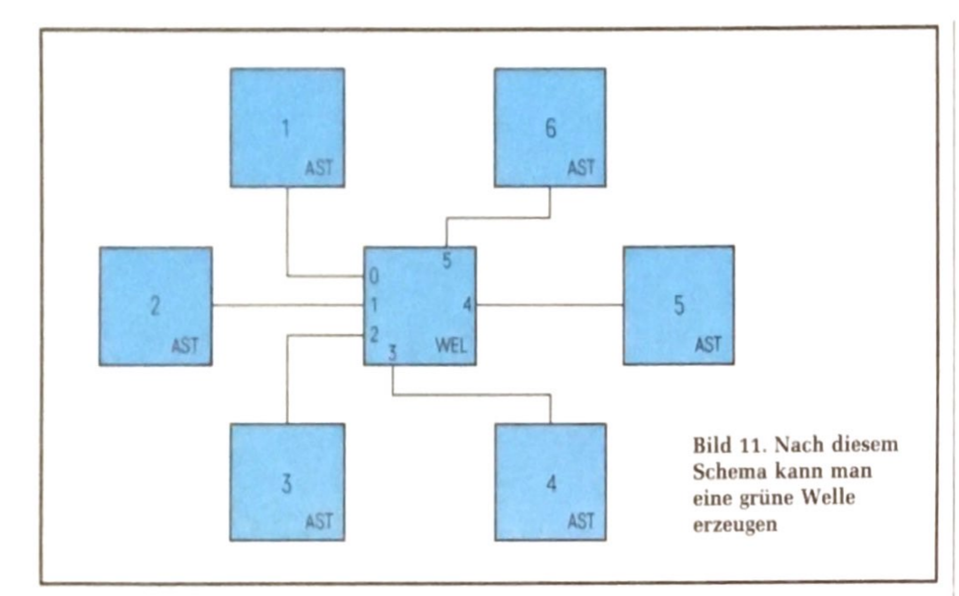


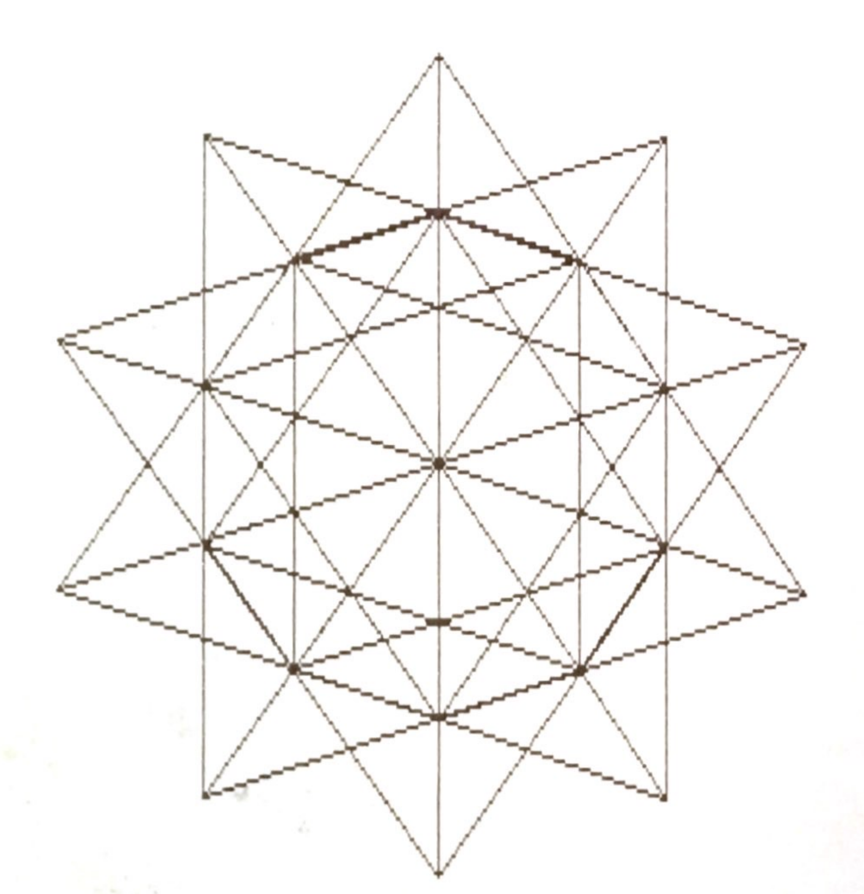
Bild 11. Nach diesem Schema kann man eine grüne Welle erzeugen

Steuerrechner wird anstelle des AST-EPROMs das EPROM WEL gesteckt. Die Ausgänge der Transistoren werden jetzt einfach an die Reset-Eingänge der SBC2-Karten geführt, parallel zur Reset-Taste liegen sie richtig. Ferner müssen natürlich alle Massen der Ampelsteuerungen miteinander verbunden werden.

Nun zur Wirkungsweise. Wenn die Spannung aller Ampelschaltungen gleichzeitig angeschaltet wird, passiert folgendes. Das Programm WEL sorgt dafür, daß zeitversetzt nacheinander die anderen SBC2-Karten durch einen kurzen Reset-Puls gestartet werden. Dadurch gibt sich ein zeitlicher Versatz bei den Ampelphasen und eine grüne Welle kann von Ampel zu Ampel laufen, wenn man die Ampeln richtig anordnet. Man kann bis zu 6 Ampelanlagen verwenden. Man kann die grüne Welle aber auch ohne das WEL-EPROM erzeugen. Dabei läßt man die einzelnen Ampelcomputer unbeschaltet und drückt nun nacheinander die einzelnen RESET-Taster der SBC2-Baugruppen, so wie die grüne Welle laufen soll. Weil unsere Computer mit einem stabilen Quarz arbeiten, bleibt ein einmal angestellter Zeitversatz auch praktisch unverändert bestehen.
In Wirklichkeit müßte die grüne Welle vom Steuerrechner in Abhängigkeit von der Verkehrslage gesteuert werden, hier ist also eine Anregung für künftige Programmier-Aufgaben, doch dazu sind wir mit unserem jetzigen Wissenstand noch nicht in der Lage.

Aufgaben

1. Was bewirkt ein Zwischenspeicher?
2. Was bewirkt der Reset-Taster bei der SBC2-Karte, bezogen auf die Ampelphasen?



```
8800:
  VIELSTAR:=$
  21 #10.W
  CD SCHLEIFE
  21 #5.W
  CD SCHLEIFE
  21 #200.W
  CD SCHREITE
  21 #144.W
  CD DREHE
  CD ENDSCHLEIFE
  21 #36.W
  CD DREHE
  CD ENDSCHLEIFE
  C9
```


Rolf-Dieter Klein

Roboter steuern

Mikroelektronik, Folge 10

In diesem Kapitel werden Sie eines der modernsten Einsatzbeispiele eines Mikrorechners kennenlernen. Mit der IOE-Platine und mit einem EPROM wird ein Roboter gesteuert werden. Sie werden sehen, daß sich hinter all diesen Dingen keine großen Geheimnisse verbergen.

Einen Roboter kann man grundsätzlich in zwei Elemente zerlegen. Ein Roboter besteht aus einer Mechanik und aus einer Steuerung.
Bild 1 zeigt das Schema. Die Mechanik kann sehr komplex sein. So kann ein Roboter aus mehreren beweglichen Armen bestehen und komplizierte Antriebe besitzen. *Bild 2* zeigt einen Industrieroboter. Die Antriebe, die nötig sind um die Arme zu bewegen, können in der Praxis sehr unterschiedlich aussehen. Es gibt hydraulische Antriebe, die mit einer Flüssigkeit arbeiten, die unter sehr hohem Druck steht und die die Antriebe bewegen kann. Dann gibt es Roboter, die elektrische Antriebe verwenden. Das können spezielle Motore sein, oft sogenannte Schrittmotore, deren Achse sich stufenweise und in Schritten drehen läßt.
Dann gibt es unterschiedliche Formen von Greifwerkzeugen. Einige Greifwerkzeuge sind zur Aufnahme von Lackierpistolen geeignet, andere können frei Gegenstände handhaben. Die Steuerung besteht meist aus einem Computer.

Roboter, handgemacht

Wenn man einen Roboter bauen will, dann sind also zwei unterschiedliche Aufgabenbereiche zu bewältigen: der mechanische Aufbau und der Aufbau der Steuerung. Unser Roboter soll folgende Aufgabe lösen: er soll Münzen vertauschen, die auf zwei Tischen A und B liegen. Als Greifwerkzeug soll ein Elektromagnet dienen. Für diesen Kurs ist von den Fischerwerken eigens ein Baukasten aus dem System Fischertechnik entwickelt worden. Mit diesem Baukasten kann man mehrere Robotermodelle aufbauen, aber auch andere „Automaten". Ausführliche Bauanleitungen liegen bei.

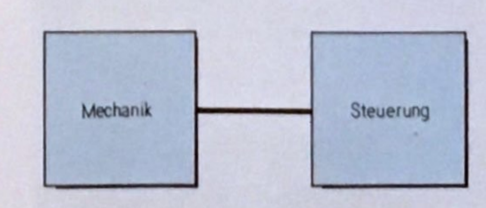


Bild 1. Steuerung und Mechanik eines Roboters verhalten sich zueinander wie Nervensystem und Körper eines Menschen

Bild 3 zeigt den prinzipiellen Aufbau unseres einfachen Roboters. Er kann sich um eine Achse drehen und er kann den Arm vor- und zurücknehmen. *Bild 4* zeigt den Baukasten.
Wir treiben unseren Roboter mit zwei Elektromotoren an. Der Computer steuert diese Motoren. Damit er weiß, wie er sie drehen muß, wird je ein Potentiometer (Poti) mit den Drehachsen des Roboters gekoppelt. Eines für den Drehkranz und eines für den Arm, wie in Bild 6 erkennbar. Von der Stellung der Schleifer der Potentiometer kann man auf die Stellung von Arm oder Drehkranz schließen. Diese Stellung wird an den Computer übermittelt.
Dazu wird das Potentiometer als Spannungsteiler geschaltet. Einer Winkelstellung entspricht dann eine bestimmte Spannung am Ausgang.

Analog – digital

Unser Computer arbeitet digital. Das bedeutet, er kann keine analogen Informationen, wie zum Beispiel die Größe der Spannung oben, direkt verarbeiten. Man muß die Spannung zuerst umwandeln. Das geschieht hier mit Hilfe eines Spannungs-Frequenz-Umsetzers. Dieser liefert am Ausgang eine Schwingung ab, deren Frequenz um so höher ist, je größer die Eingangsspannung ist. Diese Frequenz kann man mit unserem Computer messen.

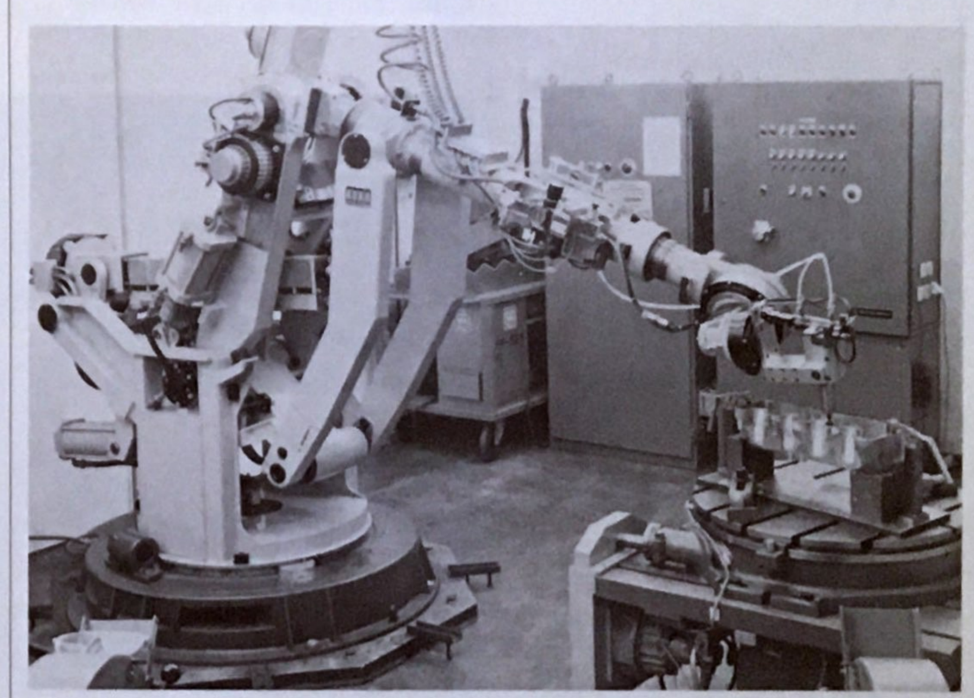


Bild 2. Ein Industrieroboter (Hersteller: Kuka) beim „Guß-Putzen", einer lärmigen und staubigen Angelegenheit

Das Meßsignal wird vom Spezial-IC 74LS627 in Form einer Rechteckschwingung geliefert, deren Periode sich ändert (im IC sind gleich zwei Umsetzer eingebaut). Man kann nun die Periode einer Schwingung einfach durch Auszählen über eine bestimmte Zeit ermitteln. So macht es unser Computer mit dem Robotersteuerprogramm.

Zum Aufbau

Man kann die Steuerschaltung (*Bild 5*) entweder auf der IOE-Karte verdrahten,

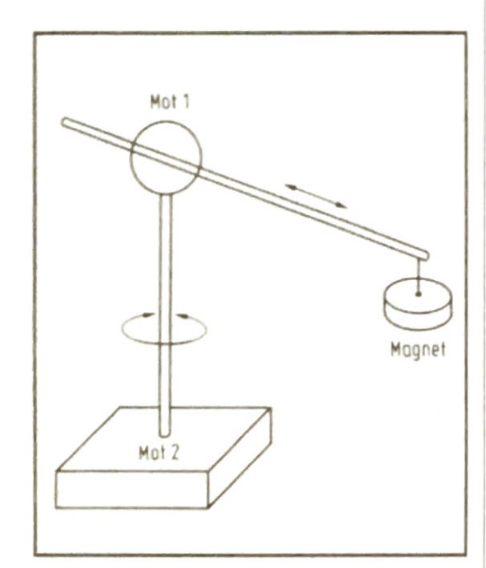


Bild 3. So ist unser Experimentier-Roboter aufgebaut

das ist mühsam, aber lehrreich, oder man verwendet die Leiterplatte ROB, die eigens entwickelt wurde und die den Rest der Schaltung faßt. Die Schaltkontakte an den 74LS245-ICs dienen der Handsteuerung. Am besten baut man sie in ein beschriftetes Steuerpult ein. Die gestrichelten Kontakte werden nicht ausgewertet. Bild 7 zeigt den Bestükkungsplan der Karte und die Anschlußbelegung des Steckers, der mit den farbigen Leitungen des Roboters verbunden wird. Beim Einsetzen der Relais ist auf die Richtung zu achten. Auf der Unterseite der Relais ist dazu eine Numerierung angegeben. Pin 1 entspricht der Markierung bei ICs.

Immer wieder: Test

Das ROB-EPROM wird in die Fassung 0 (IC 6) der SBC2-Karte gesteckt. Das Steuerprogramm benötigt RAM-Speicher. In Fassung 2 muß ein solcher Baustein stecken. Die IOE-Karte wird auf den BUS gesteckt. Auf der IOE-Karte dürfen jetzt nur die Brücken 7 und 6 eingelötet sein, damit sie sich angesprochen fühlt. Die Brücke bei e0 wird auf der Roboterkarte hergestellt, man kann sie aber auch auf der IOE-Karte setzen, auf der Lötseite der Karte, um die Stifte der Steckerleiste nicht zu verlöten. Sie schaltet das IC auf Ausgang. Die Roboterkarte (oder Ihre Eigenbausteuerung) wird zunächst noch nicht angeschlossen.

Wir messen mit dem Prüfstift bei laufendem ROB-EPROM an Pin 20 der CPU, dem IORQ-Ausgang. Die Leuchtdioden W1 und W2 müssen leuchten, die LED H brennt vorwiegend. Die LED L ist praktisch dunkel. Pin 19 der ICs 74LS245, I/O-0 und I/O-1, zeigen ebenfalls ein solches Bild. Auch an Pin 11 des ICs 74LS374, OUT 0, liegen Pulse an, wie der Prüfstift zeigt. Pin 11 von OUT 1 zeigt dagegen keine Pulse, da dieses IC nicht angesteuert wird. Nach diesen Tests kann man die Robotersteuerelektronik mit dem angeschlossenen Roboter in Betrieb nehmen.

Testschritt 1

Man betätigt nun alle Tasten der Robotersteuerung bis auf MERKE und START. Die entsprechenden Motoren für Arm und Drehkranz müssen sich bewegen. Die Drehrichtung muß sich entsprechend der betätigten Taste ändern, also zum Beispiel Drehkranz links oder rechts. Durch die EIN-Taste wird der Hub-Magnet eingeschaltet und durch die AUS-Taste wieder aus. Die angeschlossene Lampe zeigt dies zusätzlich an.

Testschritt 2

Nun sollte man kontrollieren, ob die Rückmeldung der Position über die Potis funktioniert. Man muß dazu die Ausgänge Pin 6 und 8 des ICs 74LS627 beobachten, und zwar mit einem Oszilloskop. Mit dem Prüfstift kann man hier leider keine großen Beobachtungen machen. Wenn man jetzt den Drehkranz mit Hilfe der Tasten dreht, so muß sich die Frequenz am Pin 8 ändern, wenn man den Arm bewegt, so muß sich die Frequenz am Pin 6 des 74LS627 ändern.

Testschritt 3

Nun kann man den ersten Versuch machen, den Roboter etwas beizubringen. Dazu wird die Taste MERKE gedrückt. Dann bewegt man zum Beispiel den ARM mit Hilfe der Tasten und drückt wieder die Taste MERKE. Danach bewegt man den Drehkranz und drückt erneut auf MERKE. Jetzt drücke man die Taste START.

Der Roboter sollte sich jetzt auf die alte Position wieder zu bewegen.

Vertauschte Drähte

Wenn der Roboter nicht mehr aufhört sich zu bewegen, so sollte man die An-

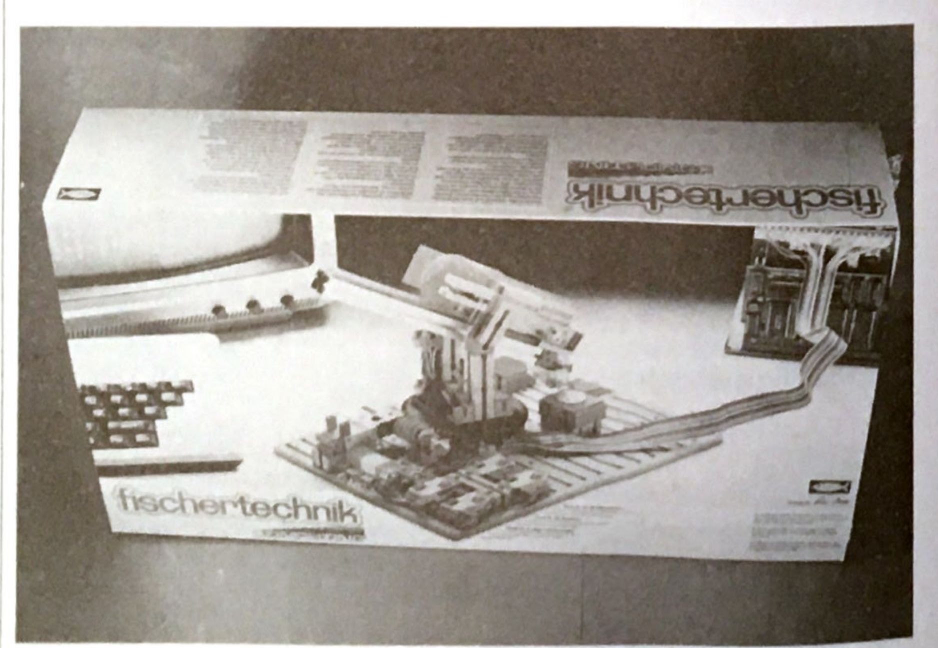


Bild 4. Auf dem Baukasten Fischertechnik Computing ist der Roboter abgebildet, wie er mit dem NDR-Klein-Computer verbunden ist

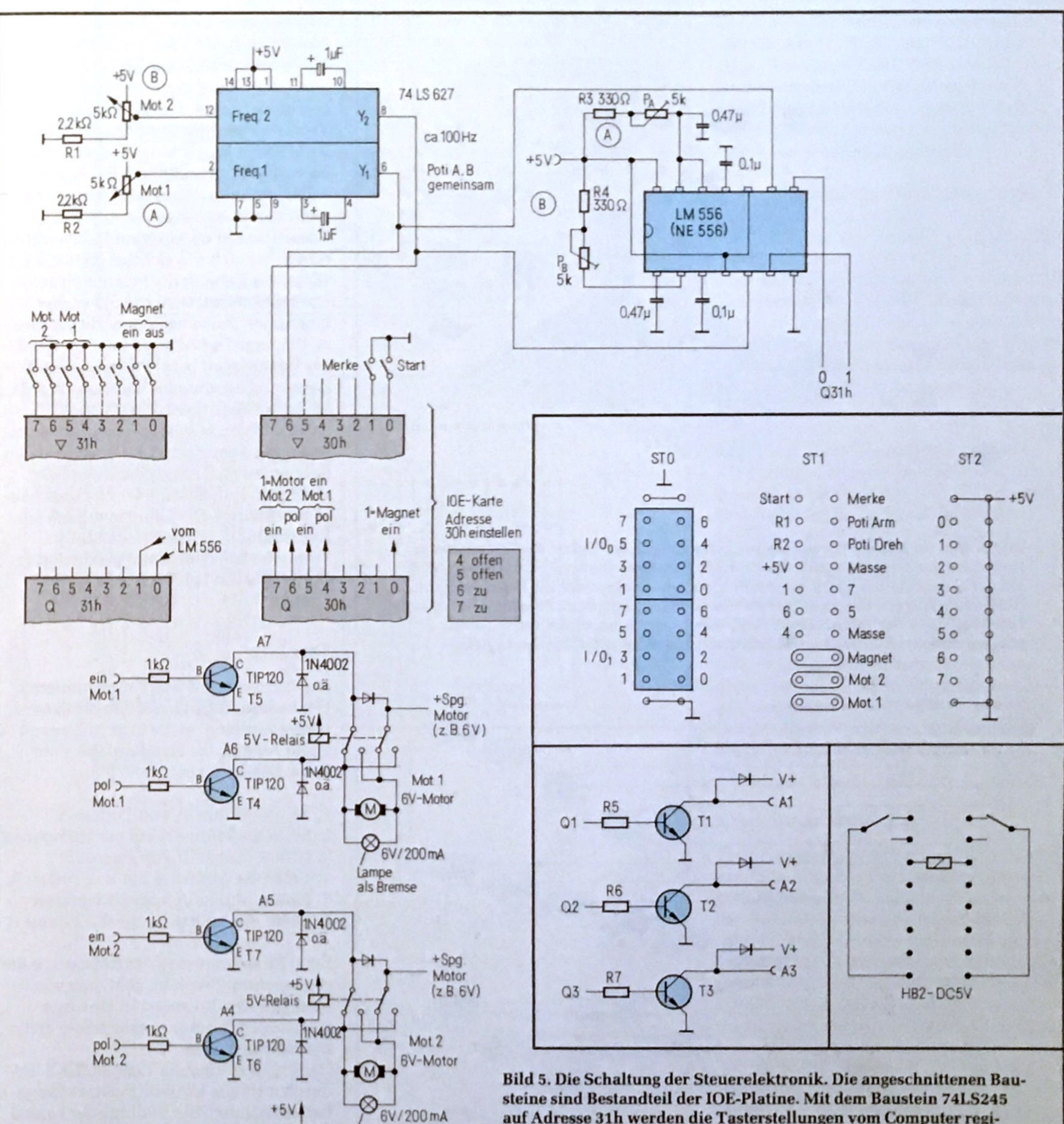


Bild 5. Die Schaltung der Steuerelektronik. Die angeschnittenen Bausteine sind Bestandteil der IOE-Platine. Mit dem Baustein 74LS245 auf Adresse 31h werden die Tasterstellungen vom Computer registriert. Über den Baustein mit Adresse 30h erfährt der Rechner die Stellung der Potentiometer, die als Frequenz vom IC 74LS627 gemeldet wird. Die Treiberschaltungen unten werden vom IC 74LS374 mit Adresse 30h angesteuert. Sie schalten die Motoren ein, um und aus. Sie betätigen auch den Hubmagneten. Rechts sind noch drei weitere Schaltverstärker und oben noch ein weiterer Wandler (IC 556) eingezeichnet, die auf der universellen Ansteuerplatine mit enthalten sind. Die Steckerbelegung dieser Platine ist ganz rechts außen angezeigt. Außerdem ist die Beschaltung des Relais HB2-DC5V angegeben. Nicht angegeben ist die Beschaltung der Steckerleiste, mit der die Platine auf die IOE-Platine gesteckt werden kann. Achtung, es darf immer nur ein Wandler-IC eingesteckt sein, sonst gibt es Schwierigkeiten; beim Roboter nur das IC 74LS627

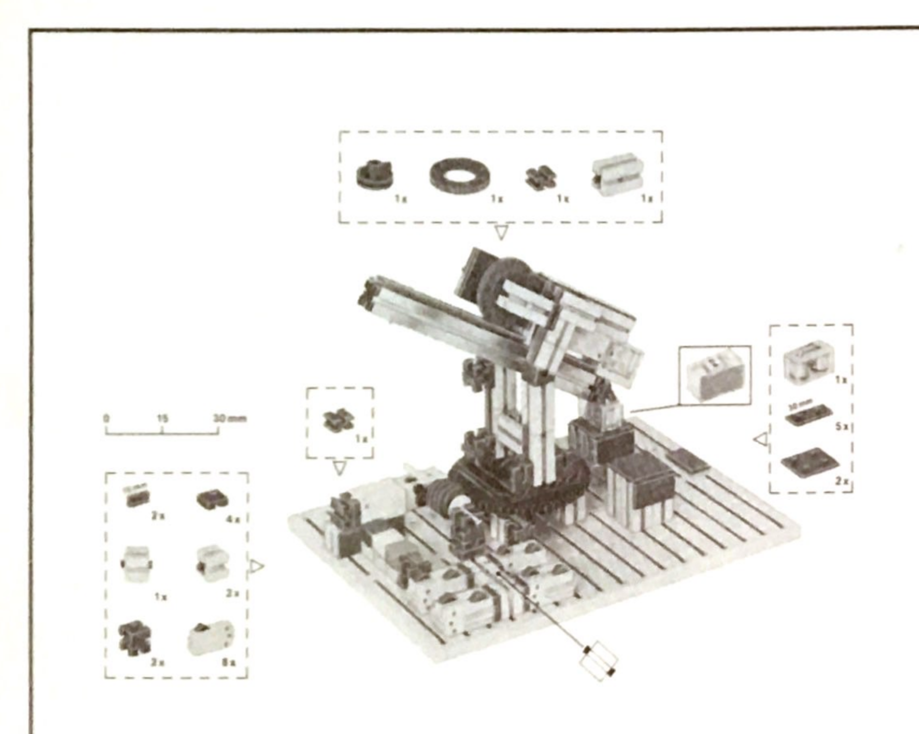
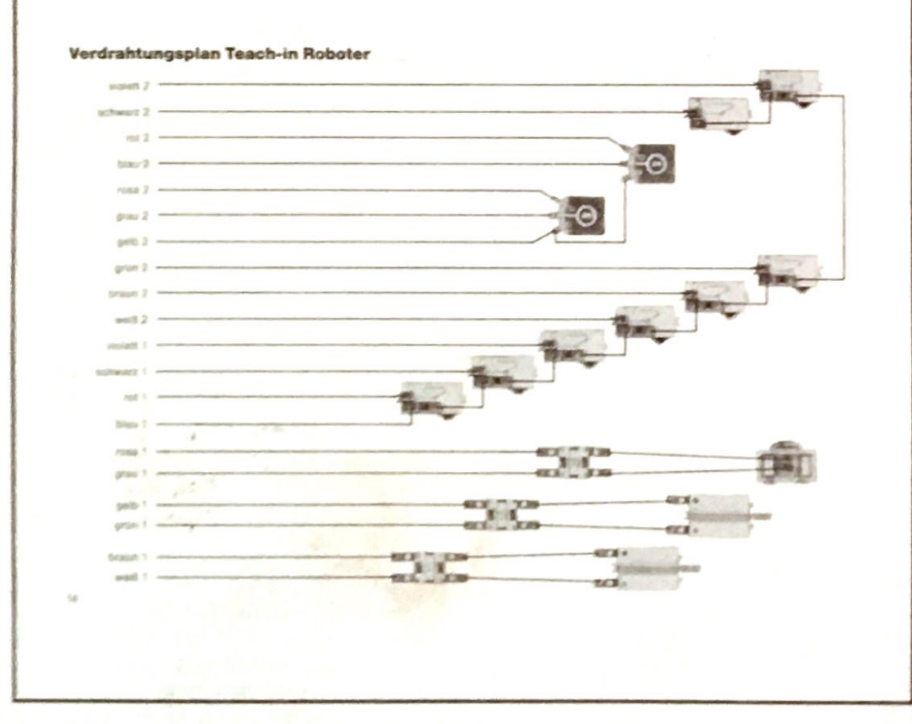


Bild 6. Auf der Roboter-Grundplatine befinden sich die Taster, die Potentiometer sind unter den Drehkränzen eingebaut, ein 20poliges Kabel verbindet den Roboter mit der Steuerplatine. Beim Verdrahten muß man aufpassen! Die Bezeichnungen stimmen, wenn von oben nach unten erst Schalter Start, dann Schalter Merke, dann Poti Arm, dann Poti Dreh, Magnet aus, Magnet ein, hebe, senke, rechts, links, Magnet, Arm und Drehkranz-Motor gezählt werden (Bild: Fischertechnik)



schlüsse des Motors, der nicht mehr stoppen will, vertauschen. Das kann bei beiden Motoren notwendig sein. Hilft diese Maßnahme nichts, so hat man vielleicht die Zuordnung vertauscht, also Motor-1-Drehkranz besitzt eine Verbindung zu Poti 2 oder umgekehrt. Dann muß man nochmals die Verdrahtung genau überprüfen.

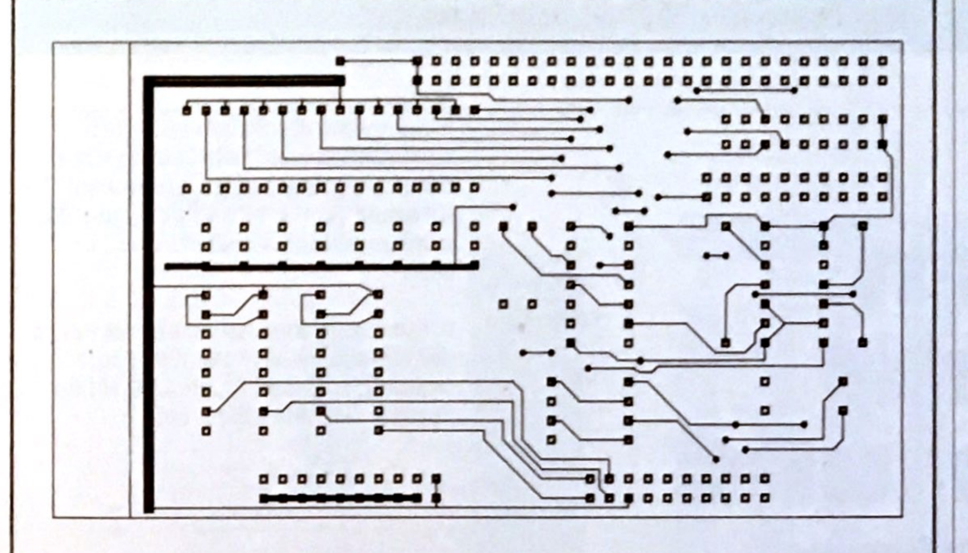
Einer dieser Fehler tritt praktisch immer auf – das liegt in der Natur der Sache bei so vielen Leitungen. Man sollte daher nicht erschrecken, sondern die Anschlüsse vertauschen. Es muß auch die Beschriftung der Tasten stimmen, also HEBE und SENKE, LINKS und RECHTS. Wenn der Motor mit falscher Polarität angesteuert wird, stimmt sie nicht. Es kann aber auch sein, daß zusätzlich die Potis falsch angeschlossen sind. Diese besitzen drei Anschlüsse die man verwechseln kann. Dann muß man jetzt sowohl die Reihenfolge der Anschlüsse am Potentiometer als auch am Motor vertauschen, bis sich die richtige Aktion zeigt. Wenn der Roboter die Positionen automatisch anfährt, wird er kurz vorher langsamer. Dabei beginnen die Lampen zu flackern. Das ist so gewollt. Es dient der Genauigkeit beim Positionieren. Die Lampen sind übrigens nicht nur Zierde, sie haben auch einen praktischen Zweck. Wenn man sie einmal probeweise aus der Fassung nimmt und dann den Roboter einen Ablauf ausführen läßt, stellt man fest, daß er jetzt viel ungenauer positioniert. Die Lampen wirken als Kurzschluß-Bremse für den Motor. Wenn der frei läuft, dann produziert er Strom, der ihn bei Kurzschluß selbst bremst.

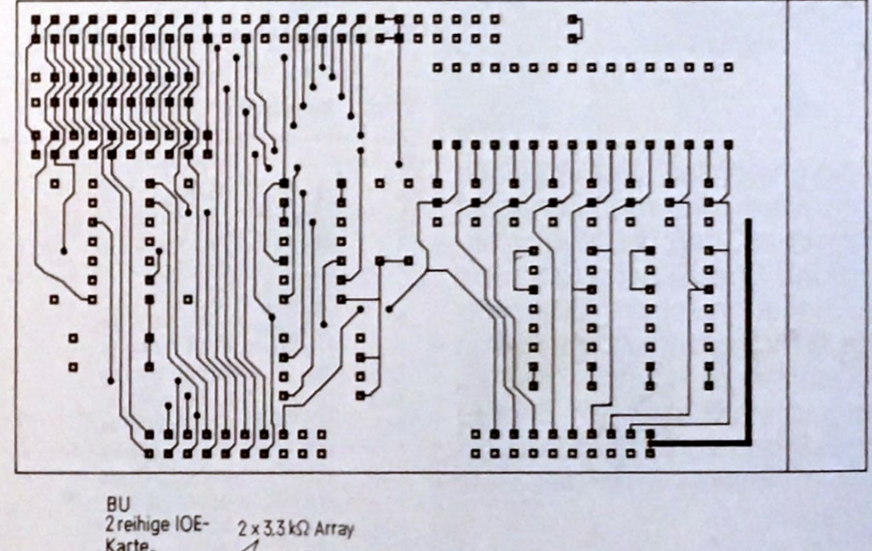
Teach In

Nun zum eigentlichen Programmieren. Um unsere Aufgabe mit den Münzen zu programmieren, sollte man sich zuerst einen Ablaufplan erstellen. Der sieht dann wie folgt aus:

1. Nimm Münze A von Podest A
2. Setze die Münze A auf ein Hilfspodest
3. Nimm Münze B von Podest B
4. Setze die Münze B auf das Podest A
5. Nimm Münze A vom Hilfspodest
6. Setze die Münze A auf das Podest B

Beim Programmieren des Roboters, beim sogenannten Teach In, geht man wie folgt vor. Er wird zuerst in eine Ausgangslage gebracht, von der aus er sich frei bewegen kann.
Dann drückt man die Taste MERKE. Dadurch wird die aktuelle Position festgehalten, nämlich die Stellung der beiden Potis und ob der Magnet aktiviert ist oder nicht.
Nun fährt man den Magnet über die Münze auf Podest A. Man betätigt wieder die Taste MERKE.
Dann drückt man die Magnet-An-Taste. Die Lampe zeigt, daß der Magnet eingeschaltet ist, die Münze muß jetzt am Magneten kleben. Es wird wieder die MERKE-Taste gedrückt.
Als nächstes hebt man die Münze mit dem Arm etwas an und drückt wieder die MERKE-Taste.





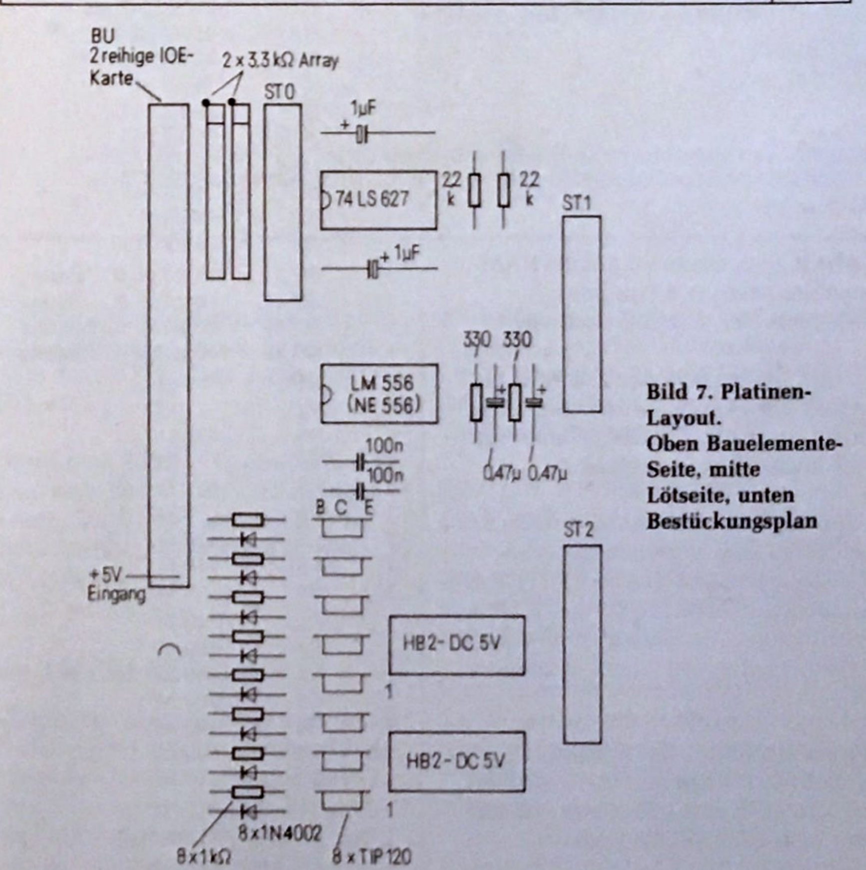


Bild 7. Platinen-Layout. Oben Bauelemente-Seite, mitte Lötseite, unten Bestückungsplan

Dann wird der Drehkranz so gedreht, daß der Arm in Richtung Hilfs-Podest zeigt. Wieder MERKE drücken.
Nun kann man den Arm zum Podest hinfahren und wieder MERKE drücken. Dann wird der Magnet ausgeschaltet und wieder MERKE gedrückt.

Jetzt sollte man das Programm erst einmal testen. Dazu drücken wir die START-Taste. Der Roboter muß jetzt die gelernte Sequenz wiederholen, also auf Startposition zurückfahren und dann alles Gelernte absolvieren. Wenn man einen Fehler gemacht hat, muß man leider wieder von vorne anfangen. Dazu drückt man die RESET-Taste der SBC2-Karte und löscht so das Bewegungs-Programm.

Genau auf diese Weise werden Industrieroboter in Teach-In-Verfahren programmiert. Bei solchen Robotern gibt es natürlich viel mehr Steuertasten um auch ganz verfeinerte Bewegungen durchführen zu können.
Roboter sind nicht nur Teach-In-programmierbar. Oft zum Beispiel gibt es eine Formel, die eine Bahn beschreibt, entlang welcher sich der Arm bewegen soll. Bei manchen Robotern kann man diese Formel direkt in den Steuerrechner eingeben.
Das Teach-In-Verfahren ist allerdings noch die häufigste Programmiermethode.

Experimentieren Sie!

Viele interessante Experimente mit dem Teach-In-Roboter sind denkbar. Man könnte versuchen, anstelle des Magnetes als Greifer, einen richtigen kleinen Greifer zu konstruieren, der mit einem weiteren Motor oder durch einen Magneten betätigt wird. Dann könnte man kleine Klötzchen bewegen und umsortieren. Oder Sie versuchen, daß Ihr Roboter sich selbst auf die Starttaste drückt. Später werden Sie in der Lage sein, auch das Roboterprogramm im EPROM selbst zu verstehen und zu verändern. Dann kann man den Roboter auf mehrere Achsen ausbauen und ihn komplizierte Tätigkeiten durchführen lassen.

Aufgaben

1. Welche Funktion hat das IC 74LS627?
2. Was passiert, wenn man die beiden Potis vertauscht?
3. Was passiert, wenn die Polung eines Motors falsch ist?
4. Was beschreibt der Ablaufplan?

Rolf-Dieter Klein

Schreiben lernen mit Tastatur und Bildschirm

Mikroelektronik, Folge 11

Vielleicht ist mancher schon ungeduldig geworden, weil er seinen Computer noch nicht richtig programmieren konnte. Zunächst aber sollten Sie ein Gefühl dafür bekommen, daß die Mikroelektronik Intelligenz in die Technik bringt und überall schaltend und regelnd eingreifen kann. Jetzt werden die technischen Voraussetzungen geschaffen, daß Ihr Computer sich „schriftlich" mit Ihnen unterhalten und Ihre Befehle entgegennehmen kann. In diesem Kapitel werden ein Tastatur-Interface und ein Bildschirm-Endgerät aufgebaut. Dieses letzte Gerät kann Ihnen vom Blümchen bis zum Schaltplan alles auf den Bildschirm malen, was das Herz begehrt. Schreiben kann es natürlich auch.

Auf der Leiterplatte GDP64K befinden sich alle Bauteile, die zum Betrieb einer Bildschirmsteuerung nötig sind. *Bild 1* zeigt die umfangreiche Schaltung, die nach und nach aufgebaut wird. Auf eine vollständige Darstellung muß hier allerdings verzichtet werden. *Tabelle 1* zeigt die Stückliste und *Bild 2* den Bestückungsplan.

So wird aufgebaut

1. Einlöten aller Fassungen. Bitte aufpassen und nicht versehentlich 14polige Fassungen anstelle von 16poligen einlöten. Man kann falsch eingelötete Fassungen praktisch nicht mehr auslöten. Dazu benötigt man entweder eine sogenannte Lötsauglitze oder eine Entlötpumpe. Also bitte vorher lieber zweimal schauen.

2. Einlöten aller diskreten Bauteile. Diskrete Bauteile sind zum Beispiel: Widerstände, Kondensatoren, Dioden, Transistoren und Quarze, also alles, was nicht mehrere Elemente integriert hat.

3. Einlöten der 36poligen Stiftleiste (Stecker 1).

4. Alle ICs einsetzen bis auf die RAM-Bausteine (4164 o. ä.) und das IC EF9366, den Grafik-Prozessor.

5. Nun kann man die Karte auf den Bus stecken. Die POW5V wird ebenfalls auf den Bus gesteckt. Die SBC2-Baugruppe wird noch nicht eingesteckt!

6. Einschalten und Messen. An Pin 8 des IC 5 (7404 beim Quarz) muß ein 14-MHz-Takt-Signal anliegen. Am Prüfstift leuchten alle vier LEDs (W1, W2, H und L). Mit einem Oszilloskop kann man diese Frequenz messen (wenn es gut genug ist).

7. Messen an Pin 1 der Fassung des EF9366 (IC 1). Dort muß auch ein Takt von 1,75 MHz anliegen. Beim Prüfstift sieht man alle LEDs leuchten.

8. Spannung abschalten und das IC EF9366 einsetzen. Dabei auf die Orientierung der Nase achten, sie muß in Richtung der ICs 19 und 21 zeigen. (Bestückungsplan!) Vorsicht, das IC ist teuer!

9. Spannung einschalten und an Pin 34 des EF9366 messen. Beim Prüfstift leuchten W1 und W2, die LED H ist dunkler und die LED L hell.

Tabelle 1. Die Stückliste zu GDP64

Stück	Benennung	
1	J1	74 05
1	J2	74 LS 166
1	J3	74 LS 163
1	J4	74 LS 00
1	J5	74 04
1	J6	4164 · 200 ns
1	J7	4164 · 200 ns
1	J8	4164 · 200 ns
1	J9	4164 · 200 ns
1	J10	4164 · 200 ns
1	J11	4164 · 200 ns
1	J12	4164 · 200 ns
1	J13	4164 · 200 ns
1	J14	74 LS 74
1	J15	74 LS 32
1	J16	25 LS 2538
1	J17	74 LS 153
1	J18	74 LS 138
1	J19	74 LS 245
1	J20	EF 9366
1	J21	74 LS 273
5	SO 14	14polige IC-Fassung
12	SO 16	16polige IC-Fassung
3	SO 20	20polige IC-Fassung
1	SO 40	40polige IC-Fassung
1	R1	75 Ω
1	R2	1 kΩ
1	R3	470 Ω
1	R4	1 kΩ
1	R5	150 Ω
1	R6	1 kΩ
1	R7	470 Ω
1	R8	330 Ω
1	R9	1 kΩ
1	C1	10 µF
1	C2	100 nF
1	C3	100 nF
1	C4	100 nF
1	C5	100 nF
1	C6	100 nF
1	C7	10 µF
1	T1	BC 107
1	Q1	14,00 MHz
1	Stecker 1	36polig
1	Platine mit Lötstopplack	

mikrocomputer schritt für schritt

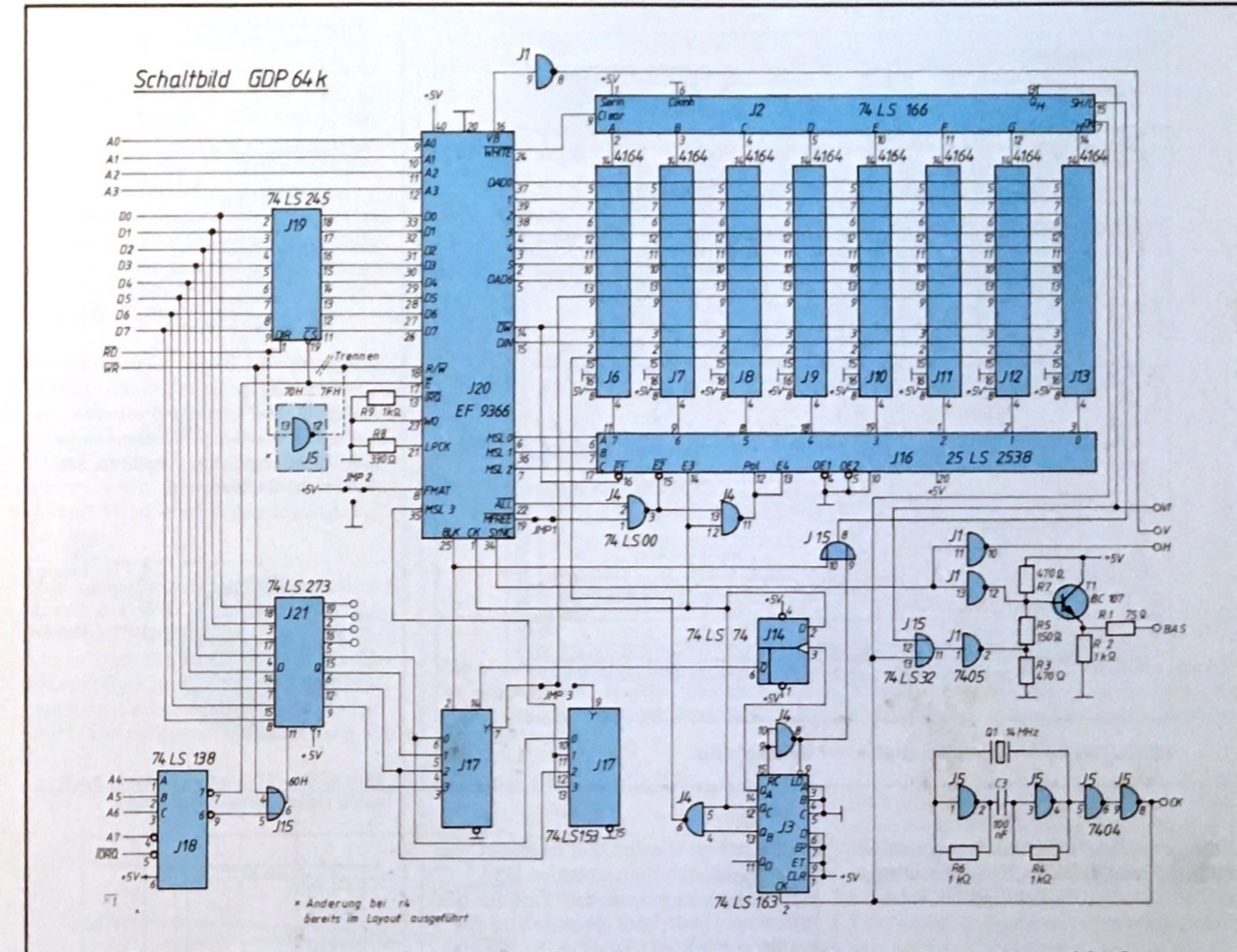


Bild 1. Der Schaltplan zur Grafikplatine zeigt viele Bauelemente und Leitungen. Keine Angst, wenn Sie sorgfältig löten, wird alles funktionieren

Mit dem Oszilloskop kann man sich die Pulsform genauer ansehen. Es handelt sich um das HSYNC-Signal, das später erklärt wird.

10. Messen an Pin 16. Mit dem Prüfstift erkennt man, daß W1 und W2 sichtbar flimmern. Die LEDs H und L leuchten auch. Es handelt sich hierbei um das VSYNC-Signal.

Daten auf den Bildschirm

Nun kann man einen Bildschirm anschließen. Dabei gibt es sehr verschiedene Möglichkeiten. Am einfachsten ist der Anschluß an einen Video-Monitor. Dieser besitzt einen BAS-Eingang, den man direkt mit dem Anschluß BAS der GDP64-Baugruppe verbinden kann. Dazu besitzt die Baugruppe am Platinenrand eine Lochreihe, deren Belegung *Bild 3* zeigt. BAS heißt Bild-, Austast- und Synchronsignal. *Bild 4* zeigt das Verbindungsschema.

Dann gibt es Fernsehgeräte mit einem AV-Eingang. *Bild 5* zeigt die Belegung eines AV-Steckers. Auch dorthin kann man das BAS-Signal direkt führen, meist ist jedoch noch ein Widerstand von 75 Ω zur Anpassung nötig.

Und dann gibt es natürlich noch die TV-Geräte, die nur einen HF-Eingang (Antenneneingang) besitzen. Und um so ein Fernsehgerät anschließen zu können, benötigt man einen HF-Modulator. Der Anschluß sieht dann wie in *Bild 6* aus. Der BAS-Ausgang der GDP64K-Baugruppe wird an den Eingang des HF-Modulators angeschlossen und der Ausgang des HF-Modulators an den Antenneneingang des TV-Gerätes. Der HF-Modulator benötigt noch eine 5-V-Spannung zum Betrieb, die an der Buchsenreihe der GDP64K-Karte herausgeführt ist.

Monitore liefern ein schärferes Bild als TV-Geräte. Der Preis von Monitoren bewegt sich zur Zeit um die 300 DM herum. Der Kauf lohnt sich, wenn man intensiver in die Mikrocomputertechnik einsteigen will. Vor allem kann man dann am Computer arbeiten, ohne den Rest der Familie vom Fernsehprogramm abzuschneiden.

Wenn man nach Anschluß eines Bildschirmgerätes wieder die Spannung einschaltet, so muß ein schwach sichtbarer

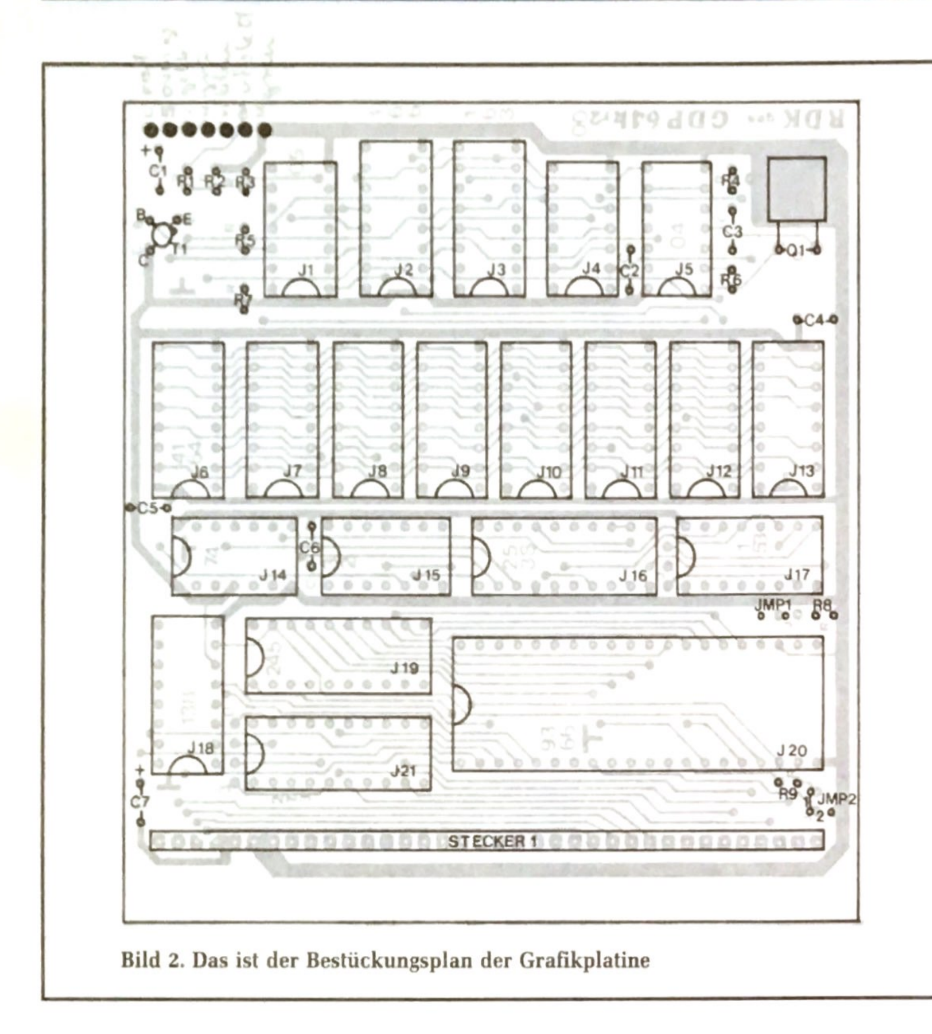


Bild 2. Das ist der Bestückungsplan der Grafikplatine

Monitor mit BAS-Eingang

BAS-Leitung

CRT-Controller

Z80-Bus

Bild 4. Schema zum Anschluß der Platine mit BAS-Eingang

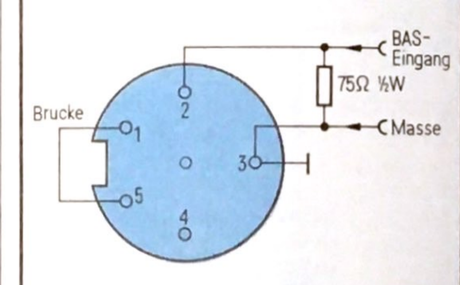


Bild 5. Die AV-Buchse verlangt nach solch einer Steckerbeschaltung

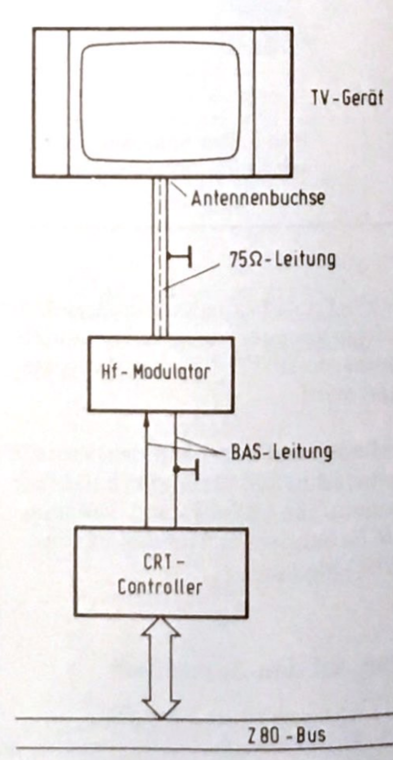


Bild 6. Mit einem Modulator wird aus dem BAS-Signal ein „Fernseh-Sender-Signal" gemacht, das jeder Fernseher empfangen kann

Rahmen auf dem Bildschirm erscheinen, abhängig von der Helligkeitseinstellung am Gerät. Dann arbeitet die GDP-Karte soweit korrekt.

Wie ein Fernsehbild entsteht

Um ein Bild auf einem Fernsehbildschirm abzubilden, muß es zunächst in Zeilen zerlegt werden. Bei unseren Geräten sind das 625 Zeilen (in den USA verwendet man eine andere Zeilenzahl). Blitzschnell wird das gesamte Bild aus Zeilen von hellen und dunklen Punkten zusammengesetzt. Auf dem TV-Gerät werden dabei zuerst alle ungeraden Zeilen eingeschrieben und nach Ablauf von 20 ms alle geraden. *Bild 7* und *Bild 8* zeigen den Ablauf. So kann man Flimmern vermeiden. Das Verfahren wird Zeilensprungverfahren genannt, da nur jede zweite Zeile geschrieben wird. Meistens wird aber bei der Erzeugung des Videosignals durch Computerelektronik das Zeilensprungverfahren nicht angewendet, sondern zweimal dasselbe „Halbbild" ausgegeben. Dadurch verringert sich der Flimmereffekt nochmals, aber man hat gegenüber einem normalen Bild nur die halbe Zeilenzahl zur Verfügung. Die GDP64-Schaltung arbeitet so.

Bei der Ausgabe zum Bildschirm genügt es nicht, die in Zeilen zerlegte Information, also das Bildsignal oder auch Videosignal genannt, einfach an den Bildschirm zu übertragen, denn der Bildschirm „weiß" ja gar nicht, wo das Bild anfängt. Dazu werden weitere Signale benötigt.

Zum einen das sogenannte Vertikal-Synchronsignal (VSYNC). Es erscheint alle 20 ms und bestimmt, wann ein Halbbild neu anfängt. Ein zweites Signal, das Horizontal-Synchronsignal, gibt an, wann eine neue Zeile beginnt. Man kann das Bild auf dem Bildschirm mit diesen Synchronsignalen sehr genau rekonstruieren. Das Horizontal-Synchronsignal, kurz VSYNC genannt, erscheint alle 64 µs.

Bild 9 zeigt eine Zusammenfassung aller Signale. Das HSYNC- und VSYNC- sowie Video-Signal werden dann noch zu einem gemeinsamen Signal gemischt, dem BAS-Signal. Die Synchronsignale kann man im BAS-Signal von den anderen Signalen durch den Spannungswert unterscheiden. Dies geschieht im Monitor oder TV-Gerät automatisch.

Für Geräte, die getrennte Eingänge besitzen, sind die Signale aber auch getrennt auf der GDP64K-Baugruppe herausgeführt.

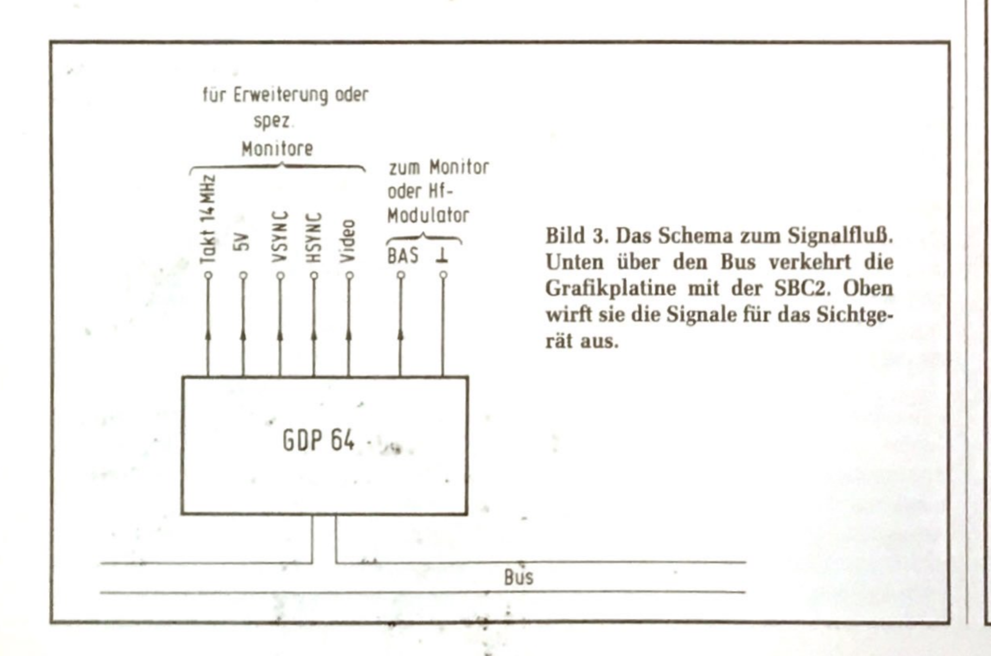


Bild 3. Das Schema zum Signalfluß. Unten über den Bus verkehrt die Grafikplatine mit der SBC2. Oben wirft sie die Signale für das Sichtgerät aus.

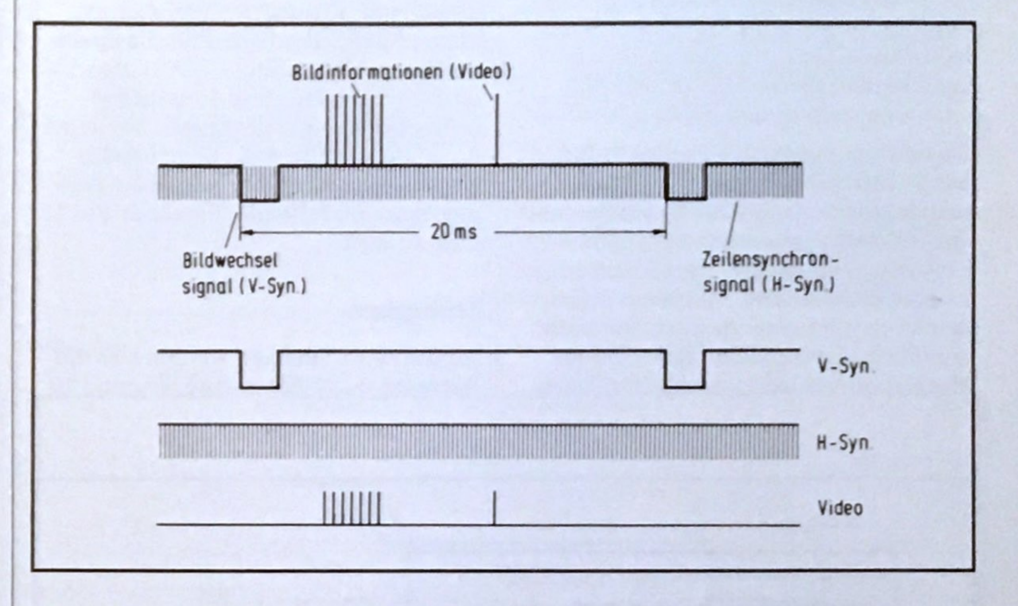


Bild 9. Das vollständige BAS-Signal ist aus der Bildinformation (Video) ganz unten, die hier aus einigen hellen Punkten in verschiedenen Zeilen besteht, und dem H-Sync-Signal, das einen jeden Zeilenwechsel bewirkt, und dem V-Sync-Signal, das dem Bildwechsel begleitet, zusammengesetzt

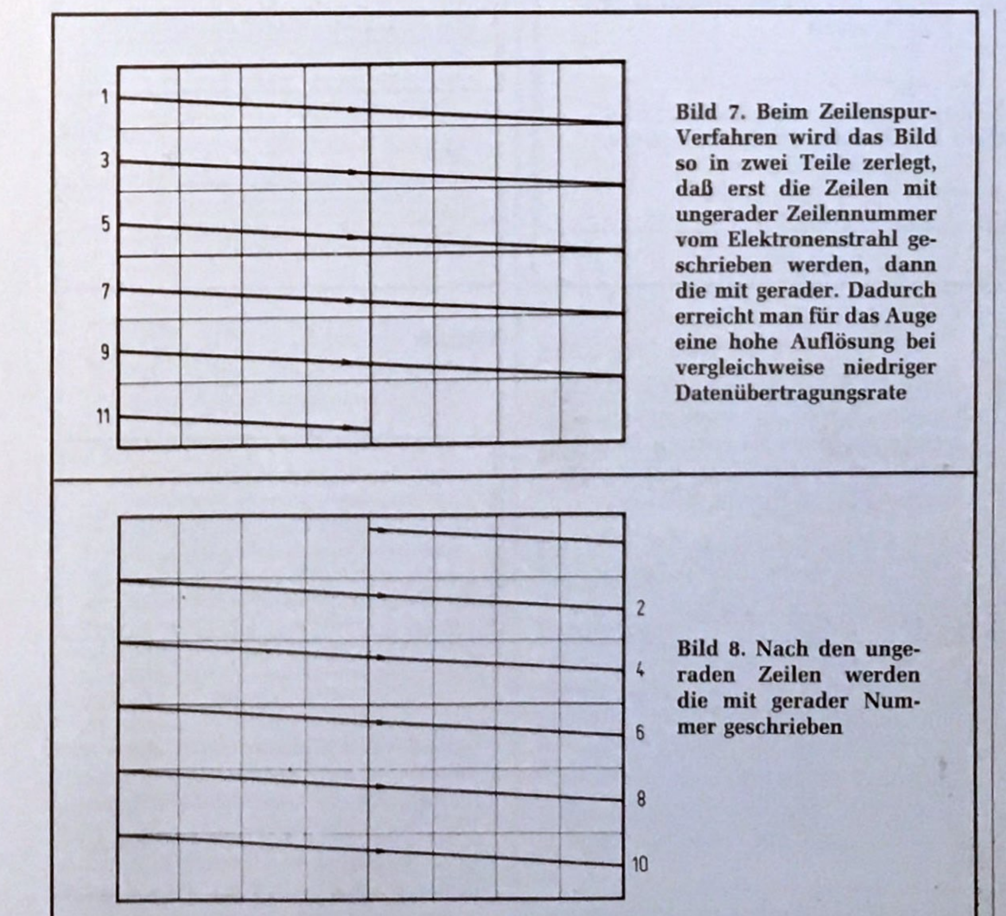


Bild 7. Beim Zeilensprung-Verfahren wird das Bild so in zwei Teile zerlegt, daß erst die Zeilen mit ungerader Zeilennummer vom Elektronenstrahl geschrieben werden, dann die mit gerader. Dadurch erreicht man für das Auge eine hohe Auflösung bei vergleichsweise niedriger Datenübertragungsrate

Bild 8. Nach den ungeraden Zeilen werden die mit gerader Nummer geschrieben

Punkte werden zu Zeichen

Ein Buchstabe, der auf dem Bildschirm erscheinen soll, muß in Rasterpunkte zerlegt werden. Diese Rasterpunkte müssen dann hintereinander so ausgegeben werden, daß die Zeichen richtig geschrieben werden (*Bild 10*). Der Abstand der Rasterpunkte ist der sogenannte Bildpunkttakt, bei uns beträgt er 14 MHz. Dieser Takt bestimmt die Auflösung, die Schärfe des Bildes, in horizontaler Richtung. Ein Bildpunkt entspricht später einem Bit einer RAM-Zelle. Ist das Bit auf 0, so leuchtet der Punkt, ist das Bit auf 1, so bleibt der Bildpunkt dunkel. Die RAM-Bausteine müssen dazu in der richtigen Weise adressiert werden. Genau diesen Job und das Erzeugen der Synchronsignale zur rechten Zeit und das regelrechte Einspeichern von Grafik-Bildpunkten und vieles mehr, das leistet alles der Baustein EF9366.

Experimente

1. Man verbinde PIN 14 des Sockels von IC 6 mit Pin 16. Dann entsteht ein Linienmuster auf dem Bildschirm (*Bild 11*). Wenn man andere Pins mit Pin 14 verbindet, ergeben sich Bilder, wie in den *Bildern 13a...13g* gezeigt. Durch Kombination dieser Signale läßt sich auch eine individuelle Zeile auswählen.

Die Informationen, welche Spalte ausgewählt wird, befinden sich auf den Leitungen DAD0...DAD6 des Grafikprozessors EF9366, die gerade verwendet wurden. Jedoch kann man diese Information so nicht sichtbar machen. Denn die Information wird gemultiplext. Das heißt, in zeitlich kurz aufeinanderfolgenden Abständen werden zwei Informationen auf denselben Leitungen übertragen. Das ist notwendig, weil die hier verwendeten RAM-Bausteine nur sehr wenige Anschlüsse haben. Das RAM besitzt zwei Steuerleitungen: $\overline{CAS}$ und $\overline{RAS}$. Die Abkürzungen bedeuten Column-Adress-Strobe (CAS), also Signal für die Spalte, und Row-Adress-Strobe (RAS), also Signal für die Reihe. *Bild 13* zeigt den zeitlichen Ablauf der Signale. Bei fallender Flanke an Pin 4 der Speicher-ICs werden die ersten acht Adreßbits übernommen. Bei fallender Flanke an Pin 15 weitere acht.

Multiplex!

An die RAM-Speicher werden also die Adressen 0...15 übertragen, das sind 16 Leitungen. Somit kann man einen Speicher von 2^{16} = 65 536 Speicherzellen adressieren. In der Baugruppe finden acht RAM-Speicher Platz, es gibt insgesamt 8 × 65 536 Bits im Speicher.

Auf dem Bildschirm werden 512 Punkte pro Zeile und 256 Zeilen dargestellt. Mit ihrem großen Speicherplatz kann die GDP64-Baugruppe vier solcher Bildebenen speichern, die man wahlweise auf dem Bildschirm sichtbar machen kann. Die Auswahl geschieht über die Leitung DAD7 (Pin 13 des Speicher-ICs), die nicht vom EF9366 herkommt, sondern von einem IC, das die Umschaltung der Seiten bestimmt. Die Aufteilung zeigt *Bild 14*.

Jeder der RAM-Bausteine ist für eine Gruppe von Linien zuständig. Jetzt können diese Bausteine eingesteckt werden. Auf die Polung achten, denn sonst werden die Bausteine zerstört. Achtung für Selbstbauer, die keine Leiterplatte verwenden! Bei den RAM-Bausteinen liegt der Masseanschluß an Pin 16 und nicht an Pin 8, wie sonst üblich. Die Spannungsversorgung liegt dagegen an Pin 8.

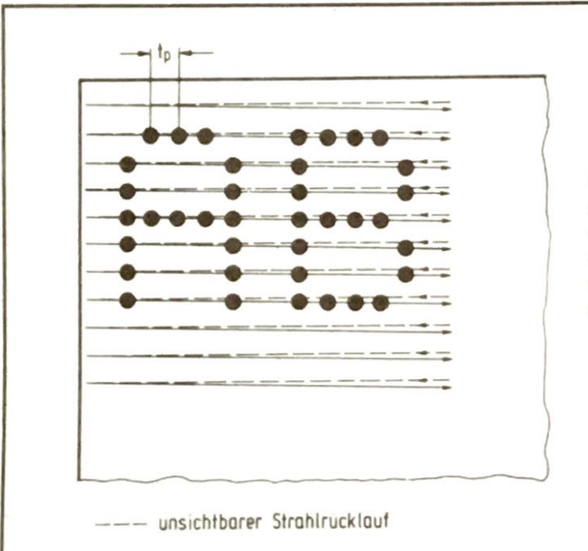


Bild 10. Ein Buchstabe ist aus Punkten zusammengesetzt, die sich in mehreren Zeilen befinden, je nach Gestalt des Zeichens

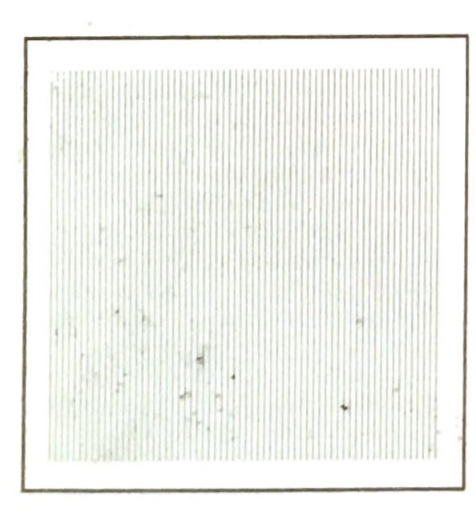


Bild 11. Das erste Testbild, das sich durch Verbinden von Pin 14 mit Pin 16 am Sockel der Speicher-ICs ergibt

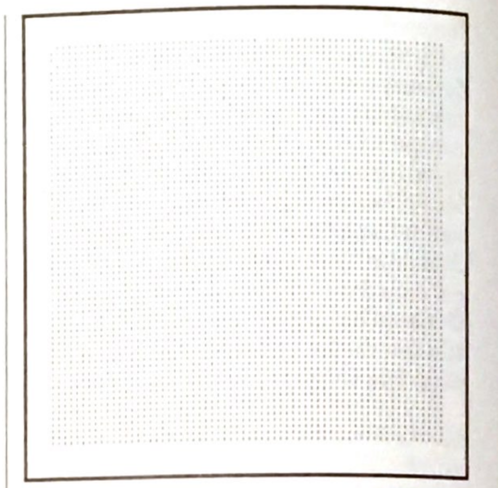


Bild 12a. Das sind die Testbilder, die entstehen, wenn man betimmte Pins am Sockel der Speicher-ICs kurzschließt: a = 14 mit 13, b = 14 mit 10, c = 14 mit 12, e = 14 mit 6, f = 14 mit 7 und g = 14 mit 5

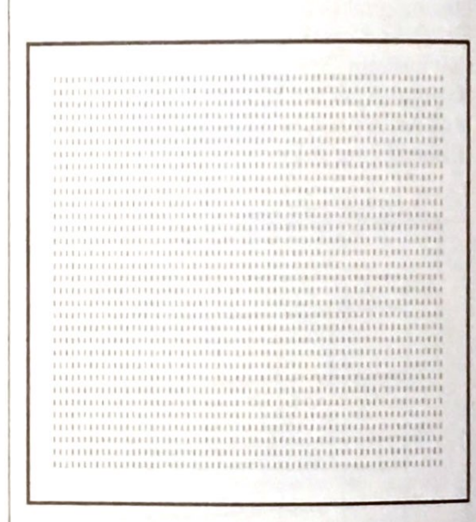


Bild 12b

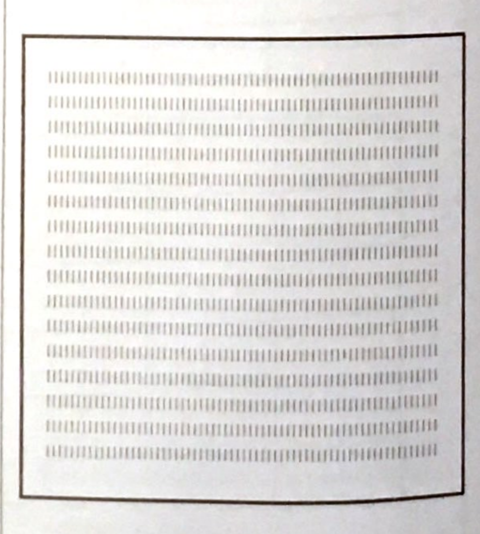


Bild 12c

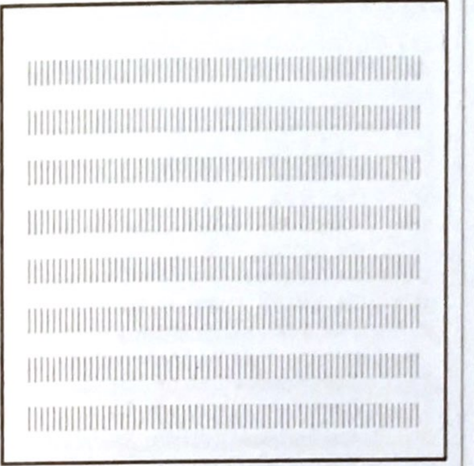


Bild 12d

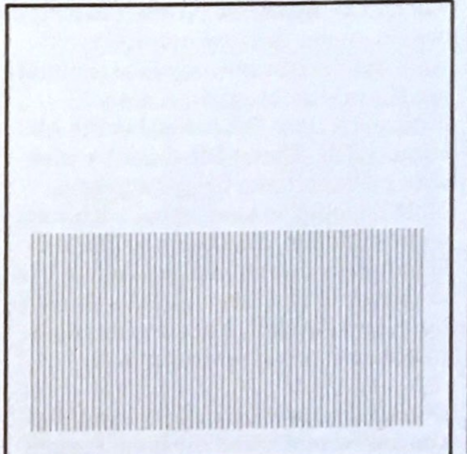


Bild 12g

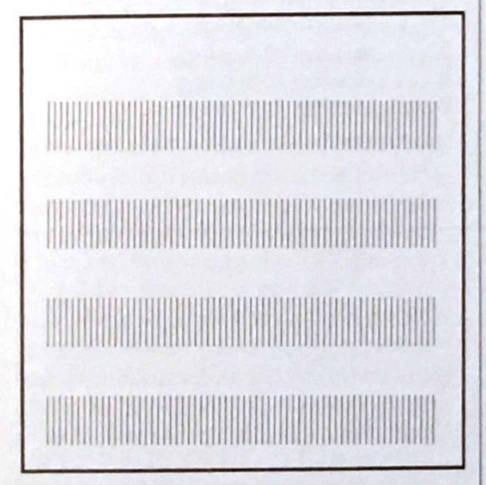


Bild 12e

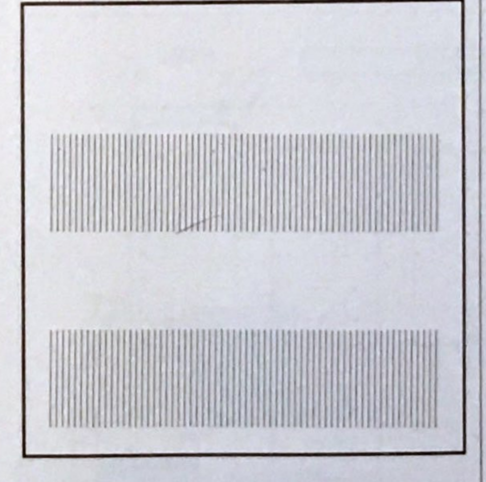


Bild 12f

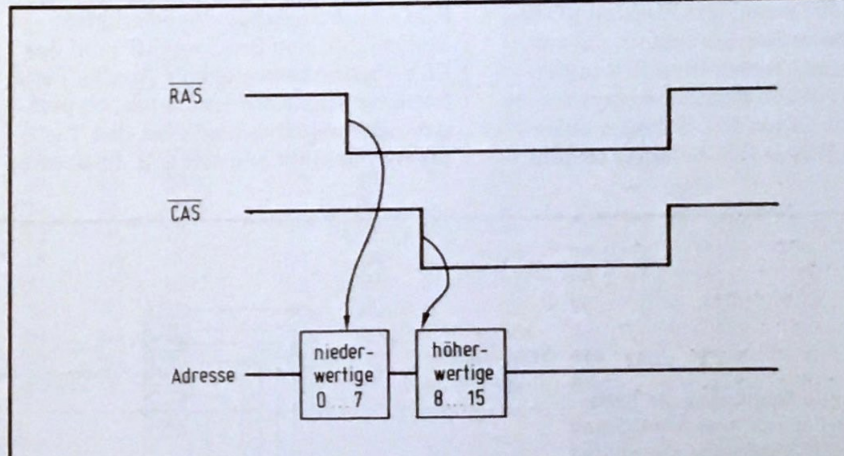
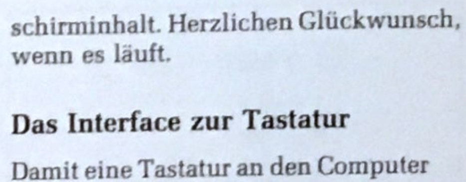


Bild 13. Die Zeitpunkte, bei welchen die Einzelteile einer Adresse ausgegeben werden, und die Signale, die eine Übernahme durch das Speicher-IC bewirken

SBC2 und GDP64

Es ist nicht möglich, Ihnen in Kürze zu erklären, wie die Platine GDP64 im einzelnen funktioniert. Es ist zunächst auch nicht wichtig – wenn Sie genau löten und nichts verkehrt einstecken, dann wird sie funktionieren. Sie werden es sehen, wenn Sie die SBC2-Baugruppe mit den zwei RAM-Bausteinen im 24poligen Gehäuse bestücken (falls sie noch nicht bei früheren Versuchen eingesteckt wurden) und das Grundprogramm in Form von zwei EPROMS 2732 einsetzen. Dabei kommt das mit 0 beschriftete EPROM in die Fassung 0 (IC 6) und das mit 1 beschriftete EPROM in die Fassung 1 (IC 7). Bitte vorher auf Pin 1 achten! Die SBC2-Baugruppe wird danach in die BUS-Karte gesteckt und die Spannung eingeschaltet. Es meldet sich dann das Grundprogramm. *Bild 15* zeigt den Bild-



schirminhalt. Herzlichen Glückwunsch, wenn es läuft.

Das Interface zur Tastatur

Damit eine Tastatur an den Computer angeschlossen werden kann, wird eine weitere Baugruppe benötigt: Die KEY-Baugruppe.

Bild 16 zeigt den Schaltplan, *Tabelle 2* die Stückliste und *Bild 17* den Bestükkungsplan. Wenn Sie den Schaltplan genau betrachten, hat er ein bißchen Ähnlichkeit mit dem der IOE-Platine. Jedenfalls gibt es Adreßleitungen, mit wel-

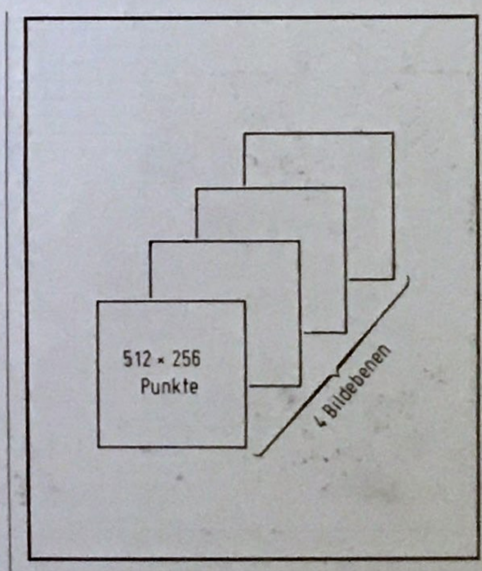


Bild 14. Das sind die vier Seiten, die voneinander unabhängige Bilder enthalten können

```
RDK-Grundprogramm

1 = aendern
2 = starten
3 = ansehen
4 = Symbole
W = weiter
```

Bild 15. Diese Meldung gibt SBC2 nach dem Einschalten ab, wenn das Grundprogramm eingesteckt ist

chen der Computer die Platine auswählt. Sie reagiert auf IORQ. Der Computer kann mit $\overline{RD}$ lesen. Das Flipflop (IC 4) und einige andere ICs helfen, die von der Tastatur hergestellten Bits in den Speicher 74LS374 einzubringen. Im Schaltplan ist ein DIL-Schalter eingezeichnet. Dieser DIL-Schalter besteht aus acht einzelnen Schaltern, die zusammen in einem Gehäuse untergebracht sind, das die Abmessung eines achtpoligen ICs besitzt. Man nennt diese IC-Form auch Dual-In-Line, daher die Abkürzung DIL. Dieser DIL-Schalter wird aber zunächst beim Grundprogramm nicht benötigt, er kann daher auch wahlweise entfallen. Man benötigt den DIL-Schalter erst, wenn man bestimmte Versuche damit machen will. Seine Einstellung wird durch IC 6 auf den Bus übertragen, wenn der Prozessor will.

Die Baugruppe wird nach Plan mit Fassungen versehen und die Bauelemente werden eingesetzt. Nun muß man noch die Verbindung zwischen Tastatur und KEY-Baugruppe herstellen. Dazu zeigt *Bild 18* die Belegung der beteiligten Stecker. Mit den Brücken JMP1 auf der KEY-Platine kann man die Art des Tastaturtaktes einstellen und auch, ob positive oder negative Logik bei den Tastatur-Bits benutzt werden soll. In unserer

Tabelle 2. Die Stückliste für KEY

Anzahl	Typ	Nr. im Schaltplan
2	74 LS 86	J1, J2
1	74 LS 04	J3
1	74 LS 74	J4
1	74 LS 00	J5
1	74 LS 245	J6
1	74 LS 374	J7
1	74 LS 32	J8
2	74 LS 85	J9, J10
3	Kondensator 100 nF	C1, C2, C4
1	Elko 10 µF	C3
1	Steckerleiste 15polig	Stecker 1
1	Steckerleiste 36polig	Stecker 2
1	Netzwerkwiderstand 8×3,9 kΩ	N1
1	DIL-Schalter 8fach	S1
2	20polige IC-Fassung	
2	16polige IC-Fassung	
6	14polige IC-Fassung	

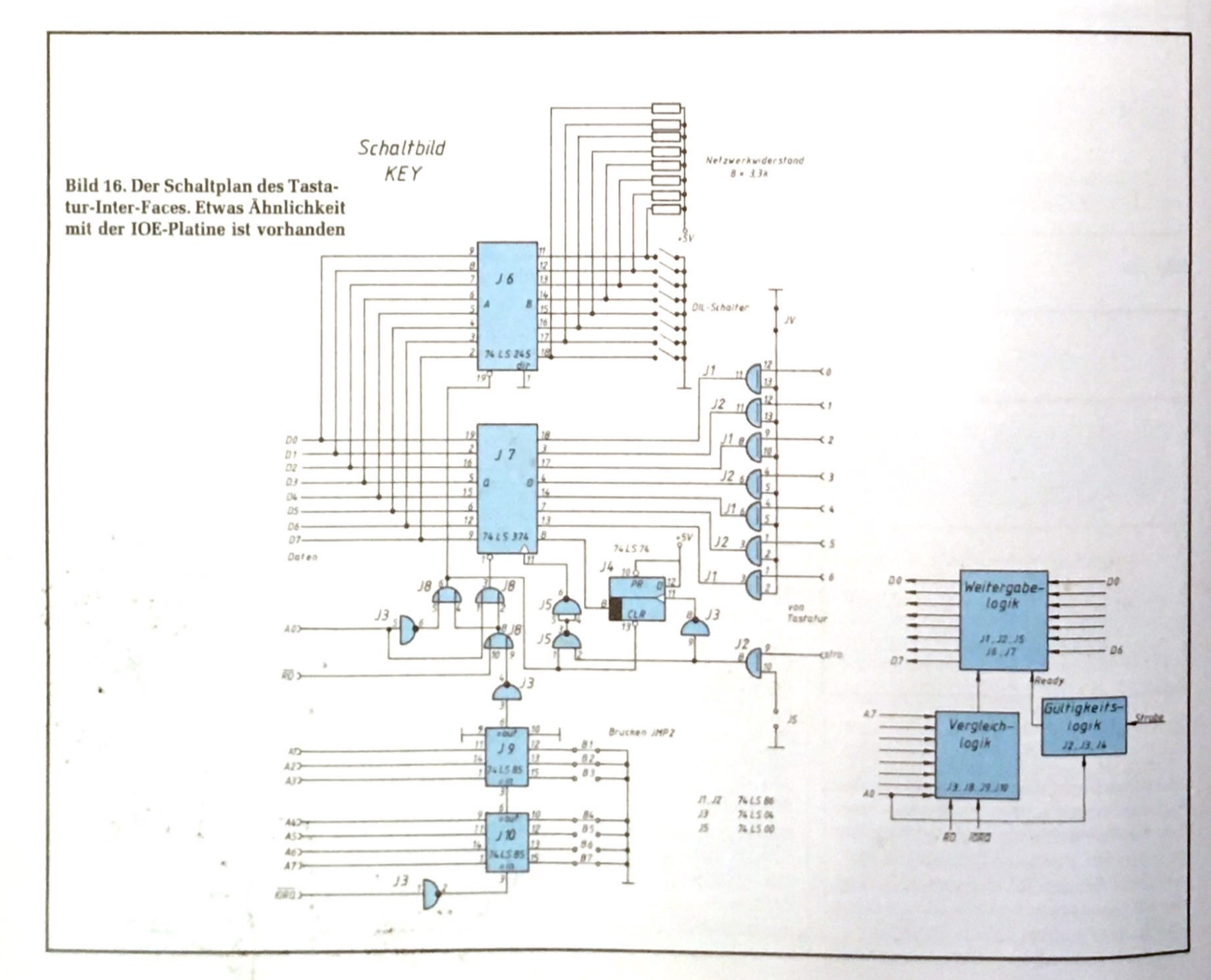


Bild 16. Der Schaltplan des Tastatur-Inter-Faces. Etwas Ähnlichkeit mit der IOE-Platine ist vorhanden

Standard-Tastatur (Cherry) befindet sich ein eigener Mikroprozessor, der die Tasteneingabe verwaltet. Er hat die Aufgabe, die Tasten zu entprellen (siehe Folge „Geschafft, er schwingt") und den Tasten duale Codes zuzuordnen.

ASCII ist Standard

Mit sieben Datenleitungen kann man 128 Tasten „codieren". Es gibt eine Standard-Zuordnungstabelle, die man ASCII-Tabelle nennt (American Standard Code for Information Interchange). Diese Tabelle ist nach DIN 66003 unter der Bezeichnung ISO-7-Bit-Code genormt (*Tabelle 3*). Darin sind drei Spalten abgebildet. Einmal Dezimalcode, dann die sedezimale Codierung (HEX) und dann das ASCII-Zeichen selbst. Neben Großbuchstaben sind auch Kleinbuchstaben, Zahlen und Sonderzeichen in der Tabelle vorhanden. Der Wertebereich von 0 bis dezimal 31 umfaßt sogenannte Steuerzeichen. Sie sollen spezielle Funktionen auslösen, wie zum Beispiel „Zeilenvorschub" (LF) oder „Wagenrücklauf" (CR) usw. Wenn man die KEY-Baugruppe auf den Bus steckt und die Tastatur anschließt, so meldet sich nach dem Einschalten wieder das Grundprogramm auf dem Bildschirm. In der linken unteren Ecke blinkt ein sogenannter Cursor. Dies ist ein helles Feld, das angibt, wo die nächste Schreibstelle liegt, wenn man Buchstaben von der Tastatur eingibt.

Tabelle 3. Die ASCII-Tabelle

dez	hex	ASCII	dez	hex	asc	dez	hex	asc	dez	hex	asc
0	00	NUL	32	20		64	40		96	60	'
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	–	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	S1	47	2F	/	79	4F	0	111	6F	o
16	10	DLE	48	30	0	80	50	p	112	70	P
17	11	DC1 XON	49	31	1	81	51	Q	113	71	q
18	12	0C2	50	32	2	82	52	R	114	72	r
19	13	DC3 XOFF	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	798	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	|	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	–	127	7F	

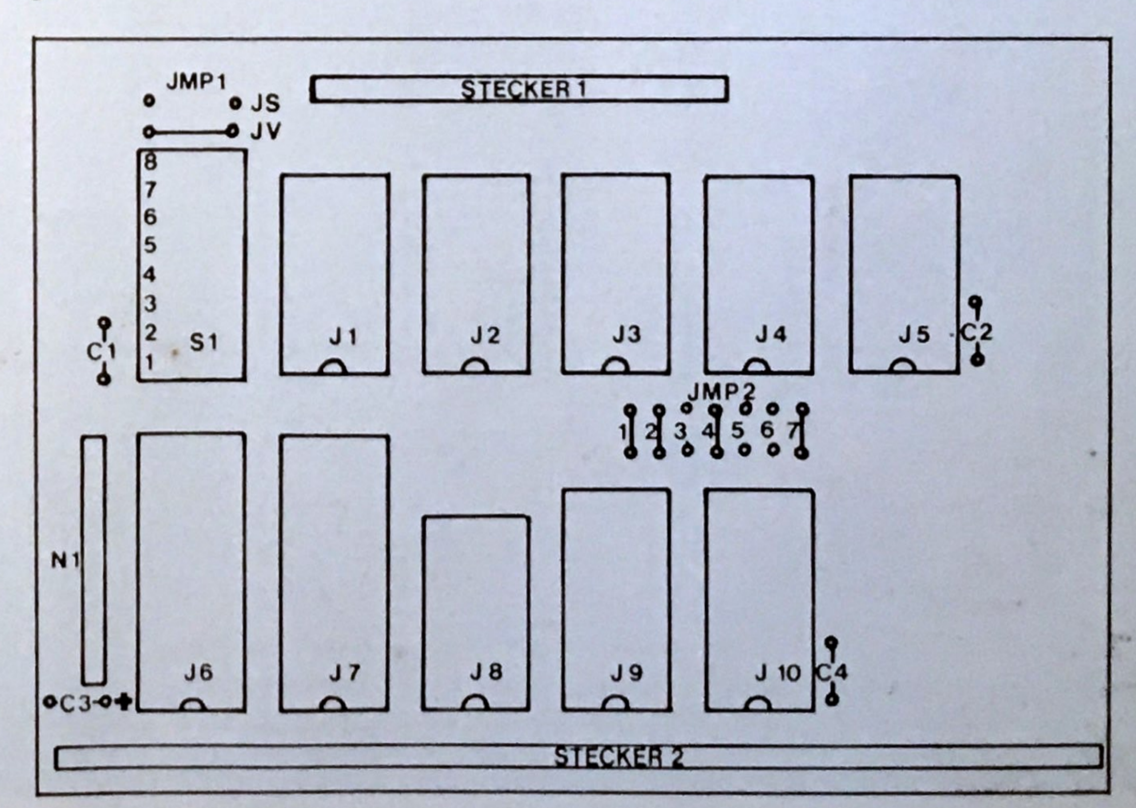


Bild 17. Das ist der Bestückungsplan von KEY

PAKETANGEBOTE

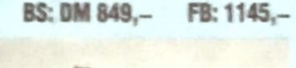
Die Paketangebote beinhalten nicht nur eine Ersparnis, sondern stellen gleichzeitig Kaufhilfen für den Noch-Nicht-Fachmann dar. Die einzelnen Baugruppen sind sinnvoll zusammengestellt und orientieren sich an der Schulfernseh-Sendereihe.

Auf Wunsch liefern wir die Teile entsprechend dem Fortschritt der Reihe jeweils kurzfristig vor dem Sendetermin. Zur Portoersparnis empfehlen wir eine Einzugsermächtigung. Bitte geben Sie beim Auftrag an, welches Programm Sie verfolgen wollen.

PAKET M — BS: DM 219,90 — FB:DM 233,–

Das kleinste Paket ist auf der hinteren Umschlagseite abgebildet. Es umfaßt den MINIBUS, eine Grundplatte speziell für diese Anordnung, das 5-V-Netzteil, die Z80-CPU mit Speicher auf der SBC 2 und die IOE/EX. Die IOE/EX enthält in festverdrahteter (gedruckter) Form den IOE-Busanschluß und alle nötigen Bauteile für die Experimente bis zur 10. Folge (Roboter). Paket M ist geeignet für alle, die zunächst nicht selbst Programme schreiben, sondern nur den Umgang mit der Hardware üben wollen. Software (EPROMs für die SBC2) muß extra bestellt werden, siehe Preistabelle.

PAKET 1 — BS: DM 849,– — FB: 1145,–




Das Z80-Paket geht den Schritt weiter zur eigenen Programmerstellung, enthält also Bildschirmanschluß und Tastatur, das nötige Grundprogramm und das Kassetteninterface. Die IOE hat das übliche Rasterfeld statt der festverdrahteten Experimente. Paket 1 besteht aus POW5V, SBC2, BUS2I, IOE, KEY, TAST, GDP64K, CAS und MON1 und deckt die ersten 15 Folgen der Sendereihe ab.

PAKET 2 — BS: DM 595,– — FB: DM 749,–

Erweitert Paket 1 zum Endausbau nach der Fernsehserie. Dazu gehören der Prozessor CPU68K, eine ROA64-Speicherbaugruppe mit dem 68008-Grundprogramm (Grafiktreiber, Editor und Assembler) und einem 8K×8-RAM-Baustein sowie der EPROM-Programmierer.

PAKET 3 — BS: DM 1435,– — FB: DM 1885,–

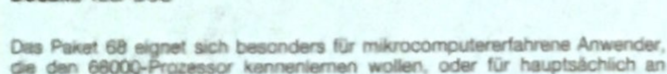
Alle Artikel aus Paket 1 und 2 zum nochmals vergünstigten Sonderpreis.

PAKET 68 — BS: DM 1950,– — FB: 2499,–

Der große NDR-Klein-Computer mit epromresidentem Assembler und PASCAL/S. Verwendet dynamische RAMs auf der DRAM128, mit gesamt 96 K nutzbarem RAM-Speicher – 32 K sind wegen Überlappung mit dem Grundprogramm ausgeblendet.

Zum Paket gehören:

CPU68K: 68008-CPU
DRAM128: 128 K RAM
ROA64: Grundprogramm
ROA64: PASCAL/S
GDP64K: Grafikinterf.
KEY: Tastaturanschl.
CAS: Kassetteninterf.
POW5V: Netzteil
TAST: Tastatur mit Gehäuse und Kabel
BUS2III: 12er-BUS



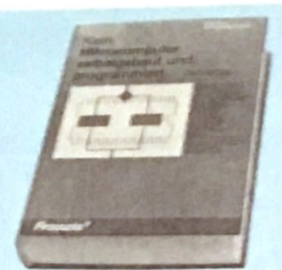
Das Paket 68 eignet sich besonders für mikrocomputererfahrene Anwender, die den 68000-Prozessor kennenlernen wollen, oder für hauptsächlich an Software-PASCAL, Grafik-Interessierte.

FLOPPY



Pakete und Geräte mit Floppy und CP/M (Warenzeichen Digital Research) sind in Vorbereitung. Fragen Sie uns bitte nach dem aktuellen Stand.

Das Foto zeigt ein Muster eines komfortablen Z80-Portable-Computers mit einem oder zwei Floppy-Laufwerken zu je 800 KByte Kapazität, mit CP/M+ (3) als Betriebssystem.



Bausätze und Geräte zum Buch und zur Schulfernsehreihe

III Mikroelektronik

GDP 64 K

Bildschirm-Steuereinheit für hochauflösende Grafik und Textanzeige. Als Bindeglied zwischen Computer und Bildschirm-Monitor zusammen mit der Tastatur Voraussetzung für die Eingabe von eigenen Programmen oder Daten. Die GDP64K besitzt Speicher für vier Bildschirmseiten (64 KByte), um schnell bewegte Grafik darstellen zu können; ist allerdings auch die teuerste Baugruppe im Paket.

ROA 64

Einfache und billige Speicherkarte für gemischte Bestückung mit EPROMs und RAMs. Dient beispielsweise der Aufnahme des 68008-Grundprogramms mit Editor und Assembler sowie der zugehörigen RAMs. Gesamtkapazität 64 KByte, die innerhalb des 1-MByte-Adreßraums frei angeordnet werden kann. Weitere ROA64s werden z. B. für die Aufnahme der PASCAL/S-EPROMs benötigt.

SBC 2

Die einzige Baugruppe, die beim Umstieg auf den 16-Bit-Prozessor überflüssig wird: Die Z80-CPU. Neben dem 8-Bit-Prozessor sind 4 K RAM und Sockel für 8 K EPROM auf der SBC2. Mit dem Z80 laufen u. a. die Versuche Musik und Ampelsteuerung, der Roboter und das Monitorprogramm für Programmierung in Maschinensprache.

PROMER

Bei Spannungsausfall sind alle Programme im RAM gelöscht. EPROMs halten dagegen den Inhalt, bis sie mit reichlich UV-Licht bestrahlt werden. EPROMs. können nur in einem speziellen Programmierverfahren beschrieben werden, wozu diese Baugruppe dient. Programmierbar sind 2716, 2732 und 2764.

CPU 68 K

Die „große" Prozessorkarte benutzt den 68008, eine Sonderversion der bekannten 68000-CPU mit 8-Bit-Datenbus. Jeder 16-Bit-Zugriff wird in zwei 8-Bit-Hälften aufgeteilt, die kurz nacheinander ausgeführt werden. Neben Geschwindigkeit sind der große Adreßbereich und klare Befehlsstruktur die wesentlichsten Vorteile des 68008 gegenüber dem Z80.

CAS

Ohne externe Speichermöglichkeit wären alle ins RAM eingegebenen Programme nach dem Stromausschalten verschwunden. Mit CAS ist es möglich, Programme oder Daten mit einem Kassettenrecorder abzuspeichern. Die Computerdaten werden in eine Serie von Einzelbits zerlegt und auf einen hörbaren Ton moduliert.

PAKET 3

Das Kombipaket (1 + 2), wie oben abgebildet, umfaßt alle in der Sendereihe benutzten Baugruppen bis auf den 21-/26-V-Wandler für den EPROM-Programmierer. Anders als auf dem Foto wird allerdings zum Paketpreis nur eine Grundplatte mit sechs Steckplätzen (BUS2II) geliefert. Wir empfehlen gegen Aufpreis den 12er-Bus (BUS2III).

Preisliste Baugruppen

Nach dem Stand vom September 1984

Typ	Funktion	P	LP	BS	FB
			Einzelpreise inkl. MwSt.		
POW5V	Spannungsversorgung 5 V/3 A	M,1,68	15,–	39,95	58,50
SBC2	Z80A-CPU, 4 K RAM, EPROM-Sockel	M,1	15,–	79,95	129,–
BUS2I	Grundpl. 6 Plätze, 4 teilbest.	1	15,–	39,95	58,50
BUS2II	Grundpl. 6 Plätze, vollbestückt	–	15,–	68,90	89,–
BU2III	Grundpl. 12 Plätze, vollbestückt	68	30,–	137,80	169,–
BU2IV	Grundpl. 18 Plätze, vollbestückt	–	45,–	206,70	249,–
MINIBUS	Grundpl. für POW5V, SBC2, IOE	M	–	19,50	24,50
IOE	16-Bit-Ein-, 16-Bit-Ausgabe	1	15,–	39,95	69,–
IOE-EX	IOE f. Exp. Musik, Ampel, Robot.	M	15,–	89,90	145,–
GDP64K	Vollgrafik m. 64-KByte-Bildsp.	1,68	15,–	359,–	458,–
KEY	Tastaturanschluß	1,68	15,–	49,95	89,–
TAST	DIN-Tastatur o. Funktionstasten	1,68	–	–	198,–
TAST/G	Gehäuse zur kleinen Tastatur	68	–	–	49,90
DINTAST	Große Tast. m. Funktionstasten	–	–	–	410,40
GEH/DT	Gehäuse zur großen Tastatur	–	–	–	112,86
TAST/K	Rundkabel für Anschluß an KEY	68	–	–	12,50
CAS	Kassettenrecorder-Anschluß	1,68	15,–	74,90	129,–
CPU68K	16-Bit-Prozessor 68008	2,68	15,–	199,–	265,–
ROA64	Speicherkarte für 8 K × 8 RAM/EPROM	2,68	15,–	39,95	84,–
DRAM64	128 K dyn. RAM-Speicher, 64-K-bestückt	–	15,–	398,–	479,–
DRAM128	128 K dyn. RAM-Speicher, vollbestückt	68	15,–	599,–	699,–
PROMER	EPROM-Programmierzusatz	2	15,–	79,95	129,–
POW21/26	Spannungswandler für PROMER	–	15,–	58,60	76,–
BANKBOOT	A16-A19-Erw. f. Z80, Floppy-Booter	–	15,–	64,50	94,–
FDC	Floppy-Steuerbaugruppe 3,5"/5,25"/8"	–	15,–	299,–	389,–
AD8 × 16	16fach-Anal.-/Dig.-Wandler 8 Bit	–	15,–	146,–	215,–
AD10 × 1	Analog-/Digitalwandler 10 Bit	–	15,–	265,–	384,–
DA	2 Digital-/Analogwandler 8 Bit	–	15,–	139,50	208,–
SER	Serielle Schnittstelle 6551	–	15,–	98,50	145,–
SOUND	AY-3-8912, Soundgenerator	–	15,–	89,90	129,–
CENT	Aufsteckplatine für IOE	–	–	10,–	20,–

P = Paketzugehörigkeit, LP = Leiterplatte, BS = Bausatz, FB = Fertigbaugruppe

Software und Begleitmaterial

Typ	Funktion	P	LP	BS	FB
			Einzelpreise inkl. MwSt.		
MON1	Monitorprogramm für SBC2	1	–	–	60,–
MUO	Musik-Testprogr. SBC2 o. RAM	–	–	–	30,–
MUM	Musik-Testprogr. SBC2 m. RAM	–	–	–	30,–
AMPEL	Programm für Experiment Ampel	–	–	–	30,–
ROBOT	Programm für Experiment Roboter	–	–	–	30,–
BASIC	BASIC für SBC2, mit Handbuch	–	–	–	75,–
GOSI	Grafiksprache f. SBC2, m. Handb.	–	–	–	75,–
MON68K	68008-Monitor/-Editor/-Assembler	2,68	–	–	155,–
PASCAL	68008-PASCAL/S	68	–	–	155,–
8K-RAM	8 K × 8 stat. RAM	2	–	–	159,–

Begleitmaterial

BUCH	R.-D. Klein: Mikrocomputer selbstgebaut u. programm.	38,–
VCAS	2 Videokassetten mit der Sendereihe (VHS, Beta, V2000)	248,–
MC-SCH	mc-Begleitheft Schaltpläne und Unterlagen	8,–
MC-Z80G	mc-Begleitheft Z80-Grundprogramme	12,–
MC-Z80A	mc-Begleitheft Z80-Aufbauprogramme	12,–
MC-68G	mc-Begleitheft 68008-Grundprogramme	12,–
MC-68A	mc-Begleitheft 68008-Aufbauprogramme	12,–
MC-PAS	mc-Begleitheft PASCAL/S-Quellprogrammliste	12,–

Zubehör

ZVM123	Bildschirm-Monitor 12", grün	293,–
ZEUG	Werkzeugsatz mit Lötkolben, Zangen, Draht usw.	69,95
BU18	Buchsenleiste 18polig für BUS2	3,80
ROBOT	Fischertechnik-Roboterbausatz	164,50
EXMUS	Bauteilsatz Experiment Musik, inkl. Eprom MUO o. MUM	41,20
EXAMP	Bauteilsatz Experiment Ampel, inkl. Eprom	34,50
EXROB	Bauteilsatz Experiment Roboter, inkl. Eprom	64,80
NETZ	Großes Schaltnetzteil 5 V/6 A, 12 V/3 A, –5 u. –12 V/0, 5 A	196,–

Alle Preise einschließlich Mehrwertsteuer.

Kostenlose Broschüre bitte anfordern!

Der NDR-Klein Computer

Die Schulfernsehserie Mikroelektronik bietet die einmalige Gelegenheit, Mikrocomputer im Detail verstehen zu lernen, mit aller denkbaren Unterstützung:

a) Fernsehsendung in 26 Folgen
b) Videokassetten davon
c) Buch von R. D. Klein
d) Begleitung durch mc und ELO
e) mc-Sonderhefte und nicht zuletzt:
f) Hardware und Anleitungen vom ELEKTRONIKLADEN

Die Sendereihe ist nicht, wie häufig, auf die Bedienung von Computern abgestellt, sondern auf das wirkliche Verstehen auch der zugrundeliegenden Elektronik. Dazu wird vom Zuschauer erwartet, daß er sein Lerninstrument von Anfang an selbst zusammenlötet und testet. Damit eignet sich der NDR-Klein-Computer nicht nur für die primären Adressaten, die Schüler weiterbildender und berufsbildender Schulen, sondern in besonderem Maße auch für Lehrwerkstätten und branchenfremde Ingenieure, die den Mikroelektronik-Zug nicht verpassen wollen.

Die „Hardware"

Der Computer wird aufgebaut in vielen überschaubaren Baugruppen, die auch der logischen Aufteilung zwischen Prozessor, Speicher und Ein-/Ausgabe (E/A) entsprechen.

Durch diese Aufteilung in Einzelbaugruppen kann der Aufbau des Computers Schritt für Schritt vorgenommen werden, sowie die Kenntnisse fortschreiten. Gleichzeitig ergibt sich dadurch große Flexibilität hinsichtlich Um- und Ausbau.

So kann der Computer auch weit über den Lehreffekt hinaus sinnvoller Verwendung zugeführt werden, wobei der finanzielle Mehraufwand für die Lernphase, z. B. bei einem bis zum CP/m-Floppy-Computer ausgebauten Gerät, schließlich vernachlässigbar wird.

BESTELLKARTE am Ende des Hefts

DRAM

Statische RAMs sind etwa vierfach teurer als die aufwendig anzusteuernden dynamischen RAMs. Bei großem Speicherbedarf lohnt sich aber der Aufwand. DRAM128 hat eine Kapazität von 128KByte und kann überlappend mit EPROMs auf ROA-Karten oder mit der Bankboot-Karte arbeiten.



AD 8 × 16

Um Temperaturen, Spannungen und andere Größen zu messen, die nicht nur Ein-/Aus-Zustände annehmen, muß ein Analog-/Digital-Wandler eingesetzt werden. AD 8 × 16 hat 16 Eingänge und unterteilt die Eingangsspannung (Bereich 0–5 V) in 256 Schritte (8 Bit). Auf dem Rasterfeld wird die Elektronik zur Umsetzung der Eingangssignale auf 0–5 V aufgebaut.



DIN-TAST

Für Vielschreiber und kommerzielle Anwendungen empfiehlt sich die große Tastatur mit Funktionstasten, Cursorfeld und Nummernfeld. Diese Sondertasten sind benutzerprogrammierbar (EPROM) und für vier Anwendungen (z. B. PASCAL, Textverarbeitung, Assembler...) umschaltbar. Ein CP/M-Programm unterstützt die Programmierung des EPROMs. Eine Schnellspeicher-Betriebsart kann 20 Zeichen zu jeder Zeit änderbar aufnehmen. Bitte Sonderprospekt anfordern.

ELEKTRONIKLADEN 4930 DETMOLD 18 ☎ 05232/8171
Verkaufsstelle München: Schulstr. 28, 8000 München 19, Tel. 0 89/1 67 94 99

ELEKTRONIKLADEN 4930 DETMOLD 18 ☎ 05232/8171
Verkaufsstelle München: Schulstr. 28, 8000 München 19, Tel. 0 89/1 67 94 99

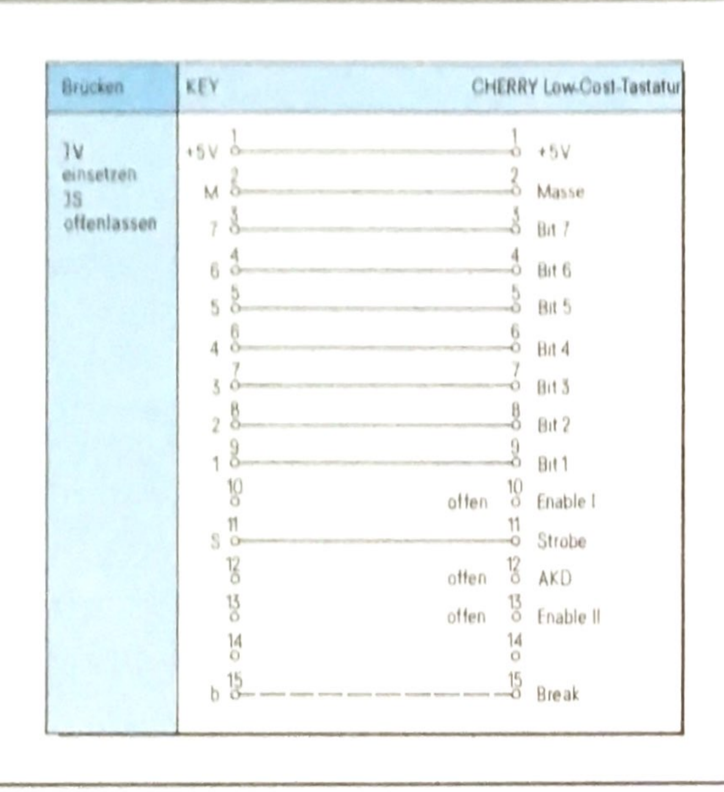


Bild 18. So wird die KEY-Platine mit der Tastatur verbunden, wenn es unsere Standard-Tastatur ist

Experimente

1. Man drückt die Taste „A" auf der Tastatur. Auf dem Bildschirm erscheint dann ein kleines „a", der Cursor wird um eine Position nach rechts verschoben. *Bild 19* zeigt den Bildschirm.

2. Wenn man erneut die Taste „A" drückt, verschwindet der Buchstabe wieder und das Cursor-Zeichen blinkt im linken Teil des Feldes.

3. Sollte nach dem Einschalten schon ein Buchstabe auf dem Bildschirm vorhanden sein, so drückt man einfach irgendeine Taste, zum Beispiel „A", und der Buchstabe verschwindet.

4. Groß- und Kleinumschaltung: Wenn man große Buchstaben eingeben will, so muß man wie bei der Schreibmaschine eine zusätzliche Taste drücken. Diese ist auf der Tastatur mit „SHIFT" bezeichnet. Dabei geht man wie folgt vor. Die Taste SHIFT drücken und den Finger drauflassen. Dann mit einem anderen Finger den gewünschten Buchstaben drücken und erst nach Freigabe der Buchstabentaste auch die SHIFT-Taste wieder loslassen.

```
RDK-Grundprogramm
1 = aendern
2 = starten
3 = ansehen
4 = Symbole
W = weiter

a
```

Bild 19. Das Anfangsmenü, das sich immer nach dem Einschalten und nach dem Reset zeigt. Von dort aus beginnt Ihr Weg in die Welt der Software und der Fähigkeiten unseres Computers

Wenn man einmal die CTRL-Taste (CONTROL-Taste) benötigt, so bedient man sie genauso als ob es eine SHIFT-Taste wäre, also vor der anderen Taste drücken und nachher loslassen. Mit der Control-Taste kann man die Codierung verändern. Allerdings ist die Art der Änderung von Tastatur zu Tastatur verschieden. In unserem Fall ergibt sich bei den Großbuchstaben zum Beispiel immer eine (in dezimal) um 64 verminderte Code-Zahl.

```
RDK-Grundprogramm
1 = aendern
2 = starten
3 = ansehen
4 = Symbole
W = weiter

w
```

Bild 20. Mit w kann man das nächste Menü, das nächste Angebot an Funktionen des Computersystems erreichen

```
1 = Laden CAS
2 = Speichern CAS
3 = Pruefen CAS
W = weiter
```

Bild 21. Das sind alles Fähigkeiten des NDR-Klein-Computers, die Sie noch kennenlernen werden

```
1 = IO lesen
2 = IO setzen
3 = Einzelschritt
W = weiter
```

Bild 22. Es geht noch weiter mit w

Auf der Tastatur gibt es auch noch ein paar Tasten mit speziellen Beschriftungen. Die Taste ALPHA-LOCK ist ein Umschalter. Wenn man sie betätigt, erscheinen alle Buchstaben als Großbuchstaben. Dabei werden aber die Ziffern normal dargestellt, die beim Drücken der SHIFT-Taste ihre zweite Bedeutung bekommen würden (die immer über der Zahl abgedruckt ist). Das gleiche gilt auch für die anderen Tasten, die eine zweite Bedeutung bei SHIFT haben. Das Zeichen auf der oberen Tastenhälfte ist nur über die SHIFT-Taste erreichbar, ALPHA-LOCK schaltet allein Buchstaben auf große Darstellung um.

```
1 = EPROM prog
2 = EPROM lesen
```

Bild 23. Nach diesem Menü würden Sie mit w wieder auf das Anfangsmenü stoßen

Deshalb gibt es auch noch eine LOCK-Taste, die wie die SHIFT-Taste wirkt, jedoch mit Feststellfunktion.

Es gibt noch eine Reihe von weiteren Steuertasten. Zum Beispiel „ESC", „DEL", „BREAK", „LINE-FEED" und „CR". Mit der Taste „CR" kann man dem Rechner mitteilen, wann eine Eingabezeile beendet ist. „CR" bedeutet „Carriage Return" oder „Wagenrücklauf". Mit der Taste „DEL" kann man ein versehentlich falsch eingegebenes Zeichen wieder löschen, denn „DEL" bedeutet „Delete" oder „Löschen". Ähnlich verhält es sich mit „BS", das bedeutet Back Space, oder Zeichen zurück. Die Taste „ESC" wird oft gebraucht, um einen Programmlauf zu unterbrechen.

Dann gibt es noch eine lange Taste. Mit dieser Taste wird, wie bei der Schreibmaschine, ein Leerraum, ein „Blank", eingegeben, um zum Beispiel Wörter voneinander trennen zu können.

Ein letzter Versuch

1. Die Taste „w" (mit einem Großbuchstaben beschriftet) wird gedrückt. Auf dem Bildschirm erscheint *Bild 20*.

2. Die Taste „CR" wird gedrückt. Damit sagt man dem Rechner, daß die Eingabe beendet ist und daß man wünscht, daß der Befehl ausgeführt werde. Die Eingabe von „w" steht für „weiter". Gemeint ist, daß das nächste Menü ausgegeben werden soll. *Bild 21* zeigt den neuen Bildschirminhalt.

3. Gibt man wieder „w" und „CR" ein, so ergibt sich *Bild 22*.

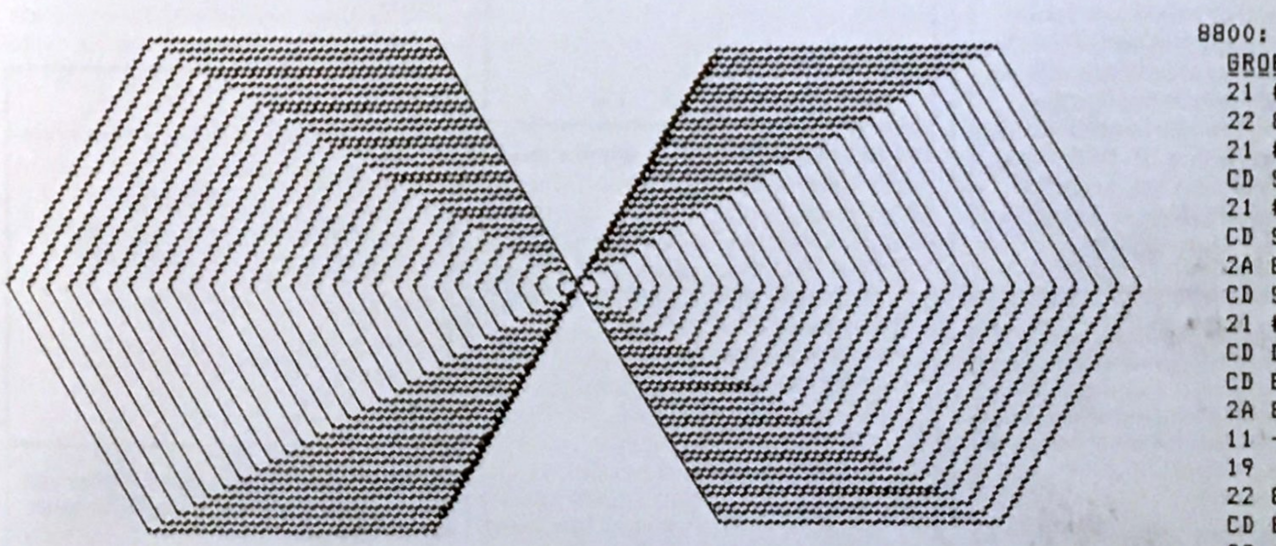
4. Und bei nochmaliger Eingabe von „w" und „CR" *Bild 23*.

Aufgaben

1. Warum muß man ein Bild in Zeilen zerlegen, wenn man es auf einem Bildschirm darstellen will?

2. Welchen Code besitzt der Buchstabe „z" gemäß der ASCII-Tabelle? Angabe in sedezimaler und dualer Schreibweise?

3. Was bewirkt die Taste „CR"?



```
8800:
  GROESSEN:=$
  21 #140.W
  22 8900.W
  21 #56.W
  CD SCHLEIFE
  21 #6.W
  CD SCHLEIFE
  2A 8900.W
  CD SCHREITE
  21 #60.W
  CD DREHE
  CD ENDSCHLEIFE
  2A 8900.W
  11 -#5.W
  19
  22 8900.W
  CD ENDSCHLEIFE
  C9
```

Rolf-Dieter Klein

Eine Sprache für den Computer

Mikroelektronik, Folge 12

Alles, was bisher im Heft geschah, sollte Ihnen etwas Gefühl für die Computerelektronik vermitteln. An mancher Stelle konnte nicht alles bis ins Letzte erklärt werden, weil dann zu viel und auch zu spezieller Stoff dargeboten worden wäre. Vielleicht ist Ihnen der Start in die Hardware aber so gelungen, daß Sie dort jetzt auch schon alleine weiterkommen. Für alle, die eher am Programmieren interessiert sind, beginnt jetzt der Teil im Heft, der in die Software hineinführt. Hardware-Vorkenntnisse benötigt man dabei nicht.

Um einen Computer sagen zu können, was er tun soll, muß man erstens wissen, was der Computer kann und zweitens wie man ihm dann das Gewünschte befiehlt.
Unser Computer kann zum Beispiel besonders gut auf den Bildschirm zeichnen. Der amerikanische Mathematiker Seymour Papert hat für solche Computer eine Sprache entwickelt, die er Schildkrötensprache nennt. Sie besteht aus besonders einprägsamen Befehlen. Sie verwendet eine Schildkröte als Symbol, weil sie in den USA eine besonders auch Kindern vertraute Figur ist. *Bild 1* zeigt, daß unsere Schildkröte auf dem Bildschirm durch ein Dreieck dargestellt ist.

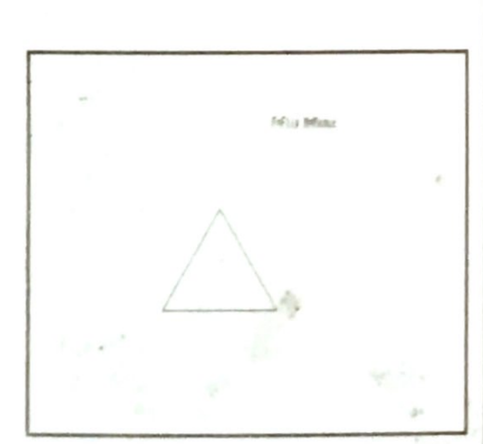


Bild 1. Ein Dreieck, das eine Schildkröte symbolisieren soll

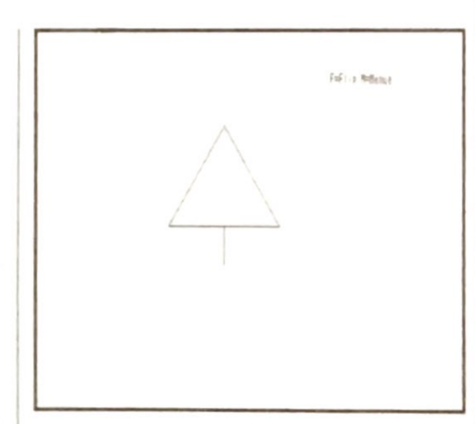


Bild 2. Die Schildkröte hinterläßt eine Spur, wenn sie schreitet



Bild 3. Die Schildkröte kann ihre Richtung ändern

Diese Schildkröte kann sich bewegen. Und zwar einmal vorwärts oder rückwärts. Dabei hinterläßt sie eine Schreibspur (*Bild 2*).
Dann kann sich die Schildkröte auch nach rechts oder links drehen (*Bild 3*). Wenn sie danach wieder schreitet, wird eine Spur in die neue Richtung gezeichnet, wie in *Bild 4* sichtbar.
Auf diese Weise kann man Bilder zeichnen. *Bild 5* zeigt ein Zehneck. Eine solche Schildkröte ist in den Computer einprogrammiert und soll einmal in Gang gesetzt werden. Dazu müssen die EPROMs mit dem Grundprogramm eingesetzt sein und dann muß der Computer eingeschaltet werden.
Er meldet sich mit dem Grundmenü (*Bild 6*).

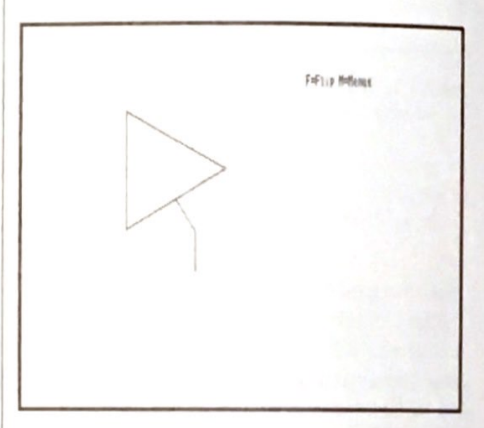


Bild 4. Mit neuer Richtung schreiten

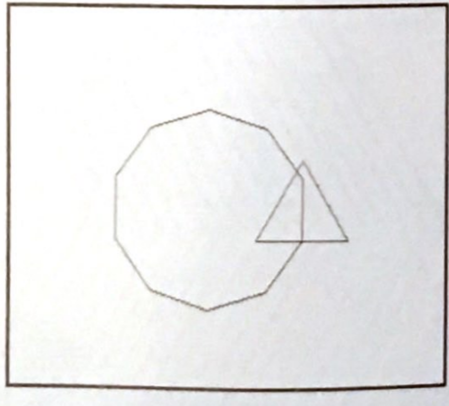


Bild 5. Ein Zehneck ist durch Schreiten und Drehen der Richtung um einen bestimmten Winkel gekennzeichnet

Das Menü erscheint

Menü bedeutet Auswahl. Es gibt eine Auswahl von Befehlen, die dem Computer jetzt gegeben werden können.


```
RDK-Grundprogramm

1 = aendern
2 = starten
3 = ansehen
4 = Symbole
W = weiter

1
```

Bild 6. Das Grundmenü. Es bietet Ihnen eine Auswahl von Kommandos an, die Sie jetzt dem Computer geben können. Die zugehörigen Aktionen sind im Grundprogramm fest eingebaut. Sie werden dazu benötigt, daß Sie mit dem Computer bequem umgehen können

Das Feld **„ändern“** bedeutet, daß man Programme oder Daten eingeben und verändern kann.
„Starten“ dient zum Starten eines Programms, nachdem es eingegeben wurde. Mit **„ansehen“** kann man sich den Inhalt eines Speicherbereichs ansehen.
Das Menü **„Symbole“** wird später zur Ausgabe von selbst definierten Namen benötigt.
Diese Befehle dienen nur dazu, den Computer bequem bedienen zu können. Sie haben mit der Schildkröte noch nichts zu tun.
Aus dem Menü benötigen wir vorerst nur die ersten beiden Befehle. Der Computer soll jetzt zum ersten Mal programmiert werden. Das geschieht mit dem Befehl „ändern“, denn jede Eingabe ändert auch irgend etwas am bisherigen Zustand. Dazu wird der Buchstabe „1“ eingetippt. Er erscheint dann links unten neben dem blinkenden Cursor-Feld. *Bild 6* zeigt das Menü mit der Eingabe.
Wenn vor der Eingabe schon ein Buchstabe zu sehen war, so kann man ihn mit der Taste „DEL“ weglöschen und danach die gewünschte Zahl eingeben.
Nun muß dem Computer gesagt werden, daß die Eingabe beendet ist. Die Taste „CR“ teilt dies in dieser Situation dem Computer mit (bei manchen Tastaturen ist die CR-Taste mit einem gewinkelten Pfeil beschriftet). CR ist die Abkürzung für „Carriage Return“ und heißt auf Deutsch: Wagenrücklauf. Der Name wurde von der Schreibmaschinentechnik übernommen, denn dort gibt es tatsächlich einen Wagen, der zurückläuft, wenn man die Taste „CR“ drückt.
Wenn man die Eingabe so beendet hat, wird der Computer den angewählten Befehl ausführen. Wenn man die CR-Taste gedrückt hat, so erscheint in unserem Fall das „Änderungsmenü“ auf dem Bildschirm. Dem Computer wurde also gesagt, daß er sich für das Ändern oder Neuprogrammieren bereit machen soll und er hat es befolgt.
Der Cursor, das ist die Marke, bei der das nächste Zeichen erscheint, wenn man eine Taste drückt, blinkt jetzt in einem Feld, das links mit „Adr:“ beschriftet ist. Der Rechner wartet jetzt auf die Angabe einer Adresse.
Programme werden im Speicher abgelegt. Ein Speicher besteht aber aus vielen Speicherzellen. Damit man eine einzelne Zelle herausfinden kann, besitzt sie eine Adresse. Diese Adresse ist eine Zahl, mit der die Speicherzellen durchnumeriert sind. Die Angabe von Speicheradressen erfolgt bei unserem Grundprogramm in sedezimaler Schreibweise. Die erste Adresse, auf der man Programme ablegen kann ist 8800. Das liegt an der Konstruktion der SBC-Karte. Man tippt also die Zahl 8800 ein und erhält *Bild 7*. Anstelle der Zahl 8800 könnte man übrigens den fest vereinbarten Namen RAM, also die Buchstaben „R“, „A“ und „M“ eintippen. Das Grundprogramm verwendet bei Nennung von RAM automatisch den ersten Speicherplatz, der frei ist.
Jetzt wird die Taste „CR“ gedrückt, um die Eingabe zu quittieren. Wenn man aber vorher einen Tippfehler gemacht hatte, so kann man falsche Zeichen von CR mit der Taste „DEL“ löschen und die richtigen Zeichen neu tippen. Der Computer wertet die Eingabe erst dann aus, wenn man die Taste „CR“ gedrückt hat. Nach der Eingabe von „CR“ erscheint *Bild 8*, wo es eine Vielzahl an Informationen gibt.
Zunächst die Angabe „8800 : 00“. Dies ist der momentane Inhalt der Speicherzelle 8800. Der Wert ist hier 0, das muß aber nicht immer so sein. Bei jedem einzelnen Exemplar unseres Computers wird etwas anderes erscheinen, denn nach dem Einschalten des Computers nehmen die Speicherzellen einen willkürlichen Wert an. Dieser Wert interessiert uns deshalb nicht.
Es wird ebenfalls noch der Inhalt der Speicherzelle 8801 ausgegeben. Auch dieser Wert ist zunächst vom Zufall eingestellt. Wir betrachten nur das breite Feld mit dem blinkenden Cursor. Dort erwartet das Grundprogramm eine Eingabe.
Im unteren Bildfeld ist eine Kurzerklärung zu sehen. Dort steht zum Beispiel „M = Menü“, gemeint ist, daß man, wenn man die Taste M drückt und mit „CR“ abschließt, wieder ins Grundmenü zurück gelangt. Oder „R = Adr“ bedeutet, wenn man die Taste „R“ drückt und dann „CR“, so kann man eine neue Adresse eingeben. Jetzt kann es ans eigentliche Programmieren gehen.

```
aendern

Adr:8800


+ =Adr + 1  - =Adr - 1
M =Menue    R =Adr
cr =ein Befehl weiter
```

Bild 7. Hat man „ändern“ aufgerufen, erscheint dieses Bild auf dem Sichtgerät. Der Computer wartet jetzt darauf, daß Sie ihn programmieren. Dabei muß der Programmbeginn festgelegt werden

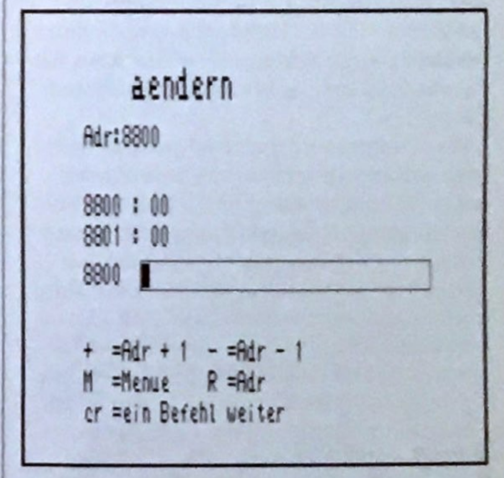


Bild 8. Der Bildschirm zeigt jetzt vierstellige Speicherzellen-Nummern in sedezimal an. Daneben nach dem Doppelpunkt auch den Inhalt. Das rechteckige Feld kann mit Befehlen an den Computer beschrieben werden

Programmieren, was ist das?

Es ist leider so, daß durch viele technische Umstände das Programmieren von Computern oft viel schwieriger erscheint, als es in Wirklichkeit sein müßte. Wie gesagt, zum Programmieren gehört ein Sack voller Fähigkeiten eines

```
aendern
Adr:8800

8800 : 00
8801 : 00
8800  21 #50.w

+ =Adr + 1  - =Adr - 1
M =Menue   R =Adr
cr =ein Befehl weiter
```

Bild 9. So sieht der Bildschirm aus, ehe CR getastet wird

Computers und eine Benennung dieser Fähigkeiten, damit man dem Computer hintereinander aufschreiben kann, in welcher Reihenfolge er seine Fähigkeiten ausüben soll. Die Schwierigkeit ist, daß ein Computer zunächst sehr merkwürdige Fähigkeiten zu haben scheint, die einem normal denkenden Menschen oft nicht einmal als nützlich erscheinen. Sie müssen sich aber vorstellen, daß die Ingenieure, die einen Computer entworfen haben, sehr genau darüber nachgedacht haben, welche Fähigkeiten, welchen Befehlssatz sie in den Computer einbauen sollen, damit man aus diesen Befehlen dann alle nur gewünschten Aktionen des Computers zusammenbauen kann.
Unser Computer ist nun so gebaut, daß er in seinem Innersten die Befehle des Z80-Mikroprozessors besitzt, denn dieser Prozessor arbeitet in unserem Computer. Der Befehlssatz des Z80 besteht aus sehr vielen verschiedenen Befehlen, die in Zahlen zwischen Null und 255 verschlüsselt sind. Dies sind alles Zahlen die gerade in ein Byte hinein passen, wenn man sie binär darstellt, wie es der Z80 auch verlangt. Unser Grundprogramm selbst gibt diese und auch alle anderen Binärwörter im Speicher sedezimal auf dem Bildschirm aus und akzeptiert von der Tastatur vorwiegend sedezimale Eingaben, wenn man den Speicher ändern will. Es darf Sie also nicht wundern, wenn die Befehle unseres Computers scheinbar sinnleere zweistellige Sedezimalzahlen sind. Es wird Ihnen alles Schritt für Schritt klar werden. Außerdem sollten Sie darüber nachdenken, daß in unserem Computer ein ziemlich umfangreiches Programm arbeitet, das Grundprogramm das es erst möglich macht, daß Sie den Computer programmieren können.

Eine Linie wird gezeichnet

Die Schildkröte soll eine Linie zeichnen. Dazu wird jetzt ein Programm eingegeben. Der erste Befehl lautet:

21 #50.W

und bedeutet: Lade den dezimalen Wert 50. Die Zahl 21 bedeutet „Lade". Sie ist der sogenannte Befehlscode, der Befehl selbst sozusagen. Der Wert „#50.W" ist die Beschreibung der Zahl, die geladen werden soll. Damit soll die Anzahl der Schritte der Schildkröte definiert werden. Das Zeichen „#" bedeutet „Dezimale Eingabe", denn der Rechner verarbeitet auch sedezimale Zahlen. Ohne das „#"-Zeichen nimmt er automatisch an, daß ihm eine sedezimale Zahl eingegeben werden soll. Der Zahlenwert ist also 50. Das „.W" ist eine Angabe des zugelassenen Wertebereichs. Es muß hier angegeben werden, denn das Grundprogramm kennt auch noch andere Zahlensorten, die später erklärt werden.
Es wird also folgendes eingetippt (*Bild 9*): Die Taste „2", dann „1", dann die lange Taste, die ein Leerzeichen erzeugt. Dann wird die Taste „#" gedrückt. Dabei muß man zuerst die SHIFT-Taste drücken und gedrückt lassen. Dann wird zusätzlich die Taste „3" gedrückt, über der sich auch das „#"-Symbol befindet. Achtung: es gibt Tastaturen, bei denen das Zeichen an einer anderen Stelle liegt. Nun wird die Taste „5" betätigt, dann „0". Achtung: Die Taste „0" ist oben bei den Zifferntasten eingereiht. Man darf sie nicht mit dem „o" verwechseln, das man bei Schreib-

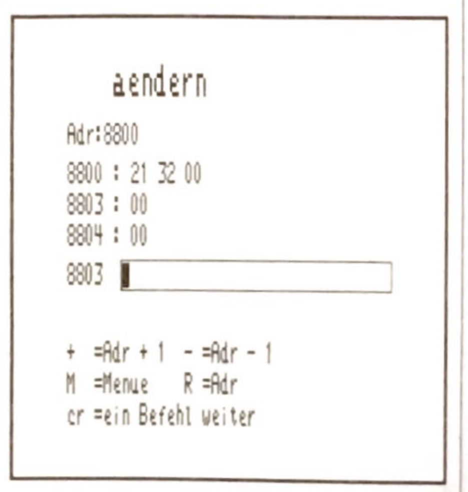


Bild 10. Es zeigt sich, daß der Computer die drei durch die Eingabe erzeugten Sedezimalzahlen hintereinander abspeichert und dann eine Eingabe für die nächste freie Speicherzelle erwartet. Was in den drei Zellen 8800, 8801 und 8802 steht ist der Befehl in Maschinencode des Z80, aber in sedezimal angezeigt

maschinen oft als 0 benutzt. Der Computer würde das nicht verstehen. Dann wird die Taste „." gedrückt und schließlich die Taste „W". Wenn auf dem Bildschirm nun ein kleines „w" erscheint, so ist das nicht schlimm, denn das Grundprogramm versteht Groß- und Kleinschreibung. Ein großes „W" erhält man, wenn man zusätzlich die Taste SHIFT drückt.

Nun tut sich noch gar nichts, klar, denn man muß jetzt die Taste „CR" als Quittung drücken, erst dann übernimmt der Rechner die Eingabe.
Der Bildschirm sieht jetzt wie in *Bild 10* dargestellt aus. Oben steht die Adresse 8800. Dann folgt:

8800 : 21 32 00

Der Inhalt der Speicherzelle 8800 ist 21. Dann folgt 32 und dann 00. Wo ist aber die Zahl 50 geblieben?
Das Grundprogramm hat die Zahl 50 in die sedezimale Zahl 32 umgewandelt (3 · 16 + 2 = 50).
Woher kommt die „00" am Schluß? Sie ergibt sich, weil der Wertebereich mit .W angegeben wurde. Der Computer kann dann beim Ladebefehl eine dezimale Zahl von +32 767 bis −32 768 auswerten. Daraus macht er eine sedezimale Darstellung von 0 bis 7FFF und von FFFF bis 8000. Die Zahl 8000 entspricht dann der dezimalen Zahl −32 768. Es wird die sogenannte Zweierkomplementdarstellung negativer Zahlen benutzt, was im Moment aber nicht weiter diskutiert werden soll, da die Umrechnung vom Grundprogramm automatisch durchgeführt wird. Die dezimale 50 ist also in 0032 umgewandelt worden, weil der Rechner intern führende Nullen mitnotiert. Und diese beiden führenden Nullen tauchen oben hinter der 32 auf, weil der Rechner gemeinerweise immer erst die beiden niederwertigen Stellen und dahinter erst die höherwertigen im Speicher ablegt. Diese Eigenart ist konstruktionsbedingt.
Bisher wurde dem Rechner noch nicht gesagt, was er mit der Zahl 50 tun soll, außer sie zu laden. Dazu muß ein weiterer Befehl eingegeben werden.
Man tippt „cd schreite", wie in *Bild 11* angegeben. Dabei kann man Groß- oder Kleinbuchstaben verwenden. Man achte darauf, daß man den Befehl „cd" von „schreite" durch ein Leerzeichen (lange Taste) trennt.
Wenn man dann „CR" drückt, so erscheint *Bild 12*. Der Code „CD" bedeutet „rufe auf". Rechts wird angegeben, was aufzurufen ist, nämlich der Befehl „schreite". Damit wird dem Rechner gesagt, er soll die Schildkröte schreiten


```
aendern
Adr:8800
8800 : 21 32 00
8803 : 00
8804 : 00
8803  cd schreite

+ =Adr + 1  - =Adr - 1
M =Menue   R =Adr
cr =ein Befehl weiter
```

Bild 11. CD ist die Codierung eines Befehles, mit dem man komplizierte Funktionen, die vorgefertigt im Grundprogramm eingebaut sind, aufrufen kann. Hier wird die Funktion schreite aufgerufen, die die vorher eingegebene Zahl nimmt und die Schildkröte auf dem Bildschirm um diese Anzahl von Schritten weitersteuert

lassen. Wie weit, das wurde durch den vorherigen Befehl, dem Lade-Wert-Befehl festgelegt.

Auf Adresse 8803 steht „CD 03 00 ! CD SCHREITE". Die linke Hälfte ist der Inhalt der Speicherzellen in sedezimaler Schreibweise, rechts, durch das Zeichen „¦" (senkrechte Linie) getrennt, steht die ursprüngliche Eingabe.

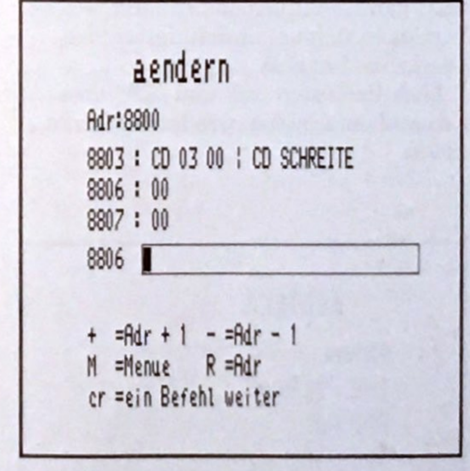


Bild 12. Das macht der Computer aus CD SCHREITE

Nun muß man dem Computer noch sagen, wann das Programm zu Ende ist. Dazu gibt es den Befehl „C9".
Die Zeichen „C" und „9" werden also eingetippt. Man kann auch hier ein kleines „c" tippen, dann erscheint auf dem Bildschirm „c9". *Bild 13* zeigt das Ergebnis. Nun noch „CR". Damit ist die Programmeingabe beendet. Nun muß man ins Menü zurück und gibt dazu den Buchstaben „M" ein. *Bild 14* zeigt das Ergebnis.

```
aendern
Adr:8800
8803 : CD 03 00 ¦ CD SCHREITE
8806 : 00
8807 : 00
8806  c9

+ =Adr + 1  - =Adr - 1
M =Menue   R =Adr
cr =ein Befehl weiter
```

Bild 13. C9 ist der Befehl, der das Ende eines Programmes anzeigt

Man gelangt aber erst dann ins Menü zurück, wenn man die Taste „CR" drückt und damit die Eingabe quittiert. Nun muß das eingegebene Programm gestartet werden. Dazu drückt man im Grundmenü die Taste „2" (*Bild 15*) und gelangt nach Eingabe ins „Startmenü" (*Bild 16*). Dort muß man die Adresse eingeben, mit der das zu startende Pro-

```
aendern
Adr:8800
8806 : C9
8807 : 00
8808 : 00
8807  m

+ =Adr + 1  - =Adr - 1
M =Menue   R =Adr
cr =ein Befehl weiter
```

Bild 14. Mit m geht es ins Menü zurück

gramm beginnt. Wir müssen da 8800 angeben. Wenn man jetzt die Taste „CR" drückt, so erscheint eine Linie auf dem Bildschirm, wie in *Bild 17*.
Das Bild zeigt nur die Linie. Auf dem Bildschirm ist in Wirklichkeit noch der

```
RDK-Grundprogramm
1 = aendern
2 = starten
3 = ansehen
4 = Symbole
W = weiter

2
```

Bild 15. Wieder das Menü. Von ihm aus wird jetzt „starten" angewählt

Pfeil der Schildkröte dargestellt und das Bild flimmert leicht. Man kann die Schildkröte ausblenden, wenn man die Taste „F" drückt. Wenn man die Schildkröte wieder einschalten will, so drücke man nochmals die Taste „F". Wenn man jetzt ins Menü zurück will, so gibt man

```
starten
Adr:ram
```

Bild 16. Ab Adresse 8800 soll losgerechnet werden

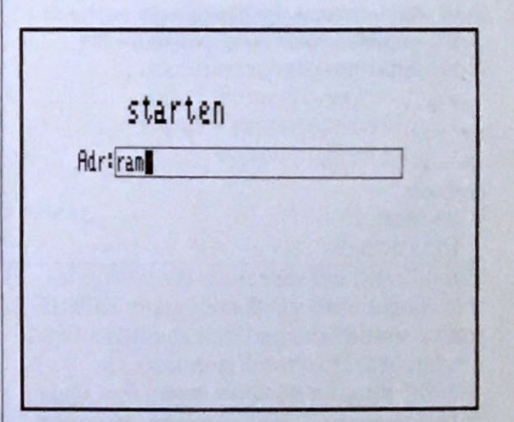


Bild 17. Das Ergebnis eines Programmlaufes

den Buchstaben „M" ein, ein Tippen von „CR" kann hierbei entfallen. Man gelangt sofort wieder ins Grundmenü zurück.
Adresse 8800, die erste freie Speicherzelle, kann man auch mit dem Namen „RAM" ansprechen. Das sieht dann wie in *Bild 18* aus.
Jetzt soll ein Quadrat gezeichnet werden. Dazu wird ein neuer Befehl benötigt. Die Schildkröte muß noch gedreht werden. Der Befehl lautet „CD DREHE". Um wieviel sich die Schildkröte dreht, hängt ebenfalls von einem Lade-Wert-Befehl ab, der davor stehen muß.



Bild 18. Man kann den Startpunkt auch mit RAM angeben, wenn man zu Beginn des RAM-Speichers starten will

Beispiel:
21 #90.W
CD DREHE
Zuerst wird der dezimale Wert 90 geladen. Dann wird der Befehl „rufe DREHE auf" ausgeführt. Die Winkel sind in Grad anzugeben. Hier wird sich also die Schildkröte um 90 Grad gegen den Uhrzeiger drehen. Gegen den Uhrzeigersinn, weil die Mathematiker die Winkel so herum zählen.

Ein Quadrat besteht aus vier Seiten. Eine Seite kann zum Beispiel durch die Folge
21 #50.W
CD SCHREITE
21 #90.W
CD DREHE
erzeugt werden. Zuerst bewegt sich die Schildkröte um 50 Punkte vorwärts, dann dreht sie sich um 90 Grad entgegen dem Uhrzeiger. Wenn man nun erneut einen Schreite-Befehl anfügt, so wird sich die Schildkröte in die neue Richtung bewegen. Wenn man also viermal die obige Sequenz aneinander fügt, so entsteht ein Quadrat. Das vollständige Programm zeigt *Bild 19*.


```
21 #50.W        21 #50.W
CD SCHREITE     CD SCHREITE
21 #90.W        21 #90.W
CD DREHE        CD DREHE
21 #50.W        21 #50.W
CD SCHREITE     CD SCHREITE
21 #90.W        21 #90.W
CD DREHE        CD DREHE
                C9
```

Bild 19. Das Programm, das ein Quadrat malt

Als letzter Befehl steht ein C9, dort ist das Programm zu Ende. Nun soll es eingegeben werden.
1. Dazu wird „1" im Grundmenü angewählt („1" tippen, dann „CR"), um das Programm einzugeben, bzw. den alten Speicherinhalt zu ändern.
2. Dann wird die Adresse eingegeben. Hier kann man wie gesagt den Namen RAM eingeben (Zeichenfolge „R", „A", „M"). Dann „CR".
3. Es erscheint der alte Speicherinhalt, wie *Bild 20* zeigt, wenn man den vorherigen Versuch gemacht hat, und zwischendurch den Rechner nicht abgeschaltet hat. Das bisherige Programm wollen wir weiter Nutzen.

```
aendern
Adr:ram

8800 : 21 32 00
8803 : CD 03 00 ; CD SCHREITE
8800 

+ =Adr + 1  - =Adr - 1
M =Menue    R =Adr
cr =ein Befehl weiter
```

Bild 20. So sieht der Bildschirm aus, wenn das erste Experiment noch nicht gelöscht ist

4. Wenn der Speicherinhalt dem Bild entspricht, so muß man diesen Teil des Programms also nicht neu eingeben, sondern kann durch Drücken der Taste „CR" den nächsten freien Speicherplatz anwählen. „CR" schaltet immer einen ganzen Befehl weiter. Man muß es zweimal tippen, dann ist man bei *Bild 21*.

5. Auf Adresse 8806 ist der Befehl „C9" zu sehen. Dieser Befehl wird nun durch einen neuen Befehl überschrieben. Nämlich durch „21 #90.W". Damit ergibt sich *Bild 22* wenn „CR" getippt wird.

```
aendern
Adr:ram
882D : CD 06 00 ; CD DREHE
8830 : C9
8831 : 00
8830 

+ =Adr + 1  - =Adr - 1
M =Menue    R =Adr
cr =ein Befehl weiter
```

Bild 21. Das zeigt sich nach zwei CRs

Auf dem Bildschirm ist dann ab Adresse 8806 die Folge „21 5A 00" angezeigt, 5A ist der sedezimale Wert für die dezimale Zahl 90.
6. Nun kann der Rest des Programms eingegeben werden. Zuerst der Befehl „CD DREHE" und so weiter bis zum „C9"-Befehl in Bild 19. Wenn man den Befehl „C9" eingegeben hat, sieht das wie in *Bild 23* aus. Der Wert auf Adresse 8831 kann bei Ihnen auch anders aussehen, da es sich um undefiniertes Speichergebiet handelt.
7. Nun die Tasten „M" und „CR" drükken und man gelangt wieder ins Grundmenü.

```
aendern
Adr:ram
8806 : 21 5A 00
8809 : 00
880A : 00
8809 

+ =Adr + 1  - =Adr - 1
M =Menue    R =Adr
cr =ein Befehl weiter
```

Bild 22. So wird weiterprogrammiert, damit ein Quadrat entsteht

mikrocomputer schritt für schritt

8. Dort die Tasten „2" und „CR" eingeben und man gelangt ins Startmenü.
9. Nun die Adresse 8800 oder den Namen RAM eingeben und die Taste „CR" drücken. Dann erscheint das Quadrat auf dem Bildschirm, wie *Bild 24* zeigt. Wenn man die Taste „M" drückt, so gelangt man wieder ins Grundmenü zurück.

```
aendern
Adr:ram
882D : CD 06 00 ; CD DREHE
8830 : C9
8831 : 00
8830 

+ =Adr + 1  - =Adr - 1
M =Menue    R =Adr
cr =ein Befehl weiter
```

Bild 23. So sieht der Schlußteil des Quadratprogrammes aus

Nun kann man sich einmal den Speicherbereich ansehen. Die Taste „3" und „CR" drücken, dann wird *Bild 25* erzeugt. Alle Speicherzellen ab Adresse 8831 haben aber wahrscheinlich einen anderen Wert als bei uns, da diese Speicherzellen ja nicht geändert wurden und beim Spannungseinschalten ein zufälliger Wert eingestellt wurde.

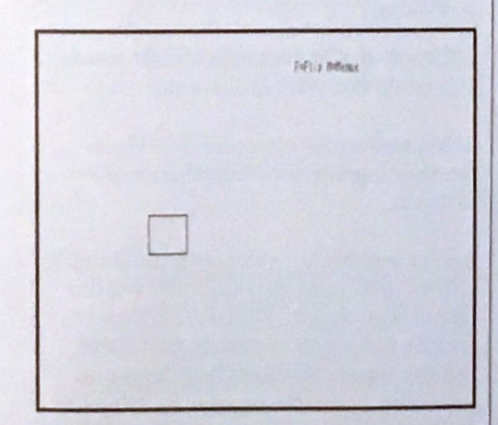


Bild 24. Das ist das Ergebnis eines Programmlaufes

Nun zwei weitere wichtige Befehle. Bisher konnte man nur geschlossene Figuren zeichnen. Es fehlte noch ein Befehl, der der Schildkröte sagt, daß sie nicht



Bild 25. Man nennt so etwas Speicherauszug. In den Zeilen stehen die Speicherinhalte in sedezimaler Darstellung. Zwischen den Zeilen sind noch die Zeichen zu sehen, die der darüberliegende Speicherinhalt ergeben würde, wenn man ihn als ASCII-Zeichen interpretiert. Nicht alle Byteinhalte ergeben ein sichtbares Zeichen auf dem Bildschirm

21 xxxx.W	Lade-Wert-Befehl
CD SCHREITE	Rufe-Schreite-Befehl
CD DREHE	Rufe-Drehe-Befehl
CD HEBE	Rufe-Hebe-Befehl
CD SENKE	Rufe-Senke-Befehl
C9	Ende eines Programms

Bild 26. Die Liste der Befehle unserer kleinen Grafiksprache. Man kann daraus schon recht leistungsfähige Programme zusammenbauen

mehr zeichnen soll. Das geschieht mit dem Befehl „CD HEBE". Und wenn sie wieder fortfahren soll zu zeichnen, so kann man das mit dem Befehl „CD SENKE" erreichen.

Die vollständige Liste der benutzten Befehle zeigt *Bild 26*.

Schreiben

Als nächstes ein paar Anregungen für weitere Figuren, die man mit dem jetzigen Befehlssatz zeichnen kann.
Sie haben sich vielleicht gefragt, wie Buchstaben auf dem Bildschirm erzeugbar sind. Hier eine Lösung mit der Zeichen-Sprache. Dazu das Programm in *Bild 27*. Man gibt es, beginnend ab Adresse 8800 ein und startet es auch mit

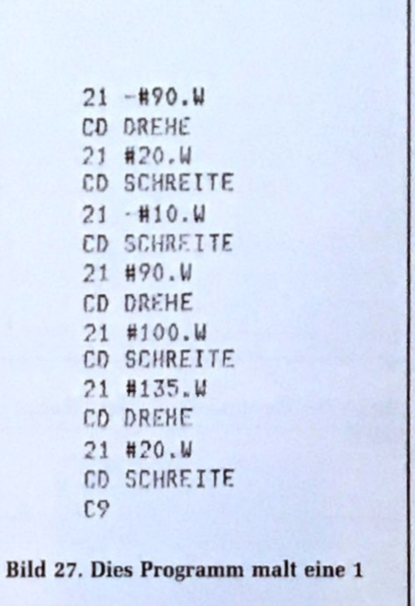
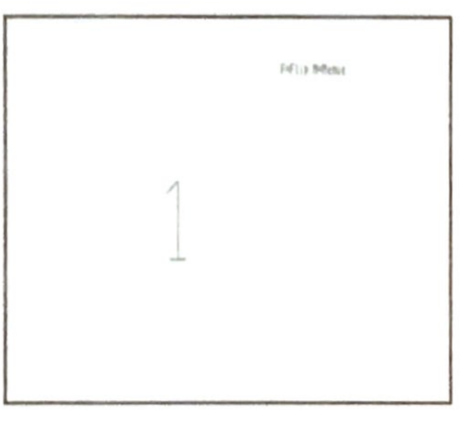

```
21 -#90.W
CD DREHE
21 #20.W
CD SCHREITE
21 -#10.W
CD SCHREITE
21 #90.W
CD DREHE
21 #100.W
CD SCHREITE
21 #135.W
CD DREHE
21 #20.W
CD SCHREITE
C9
```

Bild 27. Dies Programm malt eine 1

dieser Adresse. *Bild 28* zeigt das Ergebnis. Die Ziffer „1" wird auf den Bildschirm gezeichnet. Im Programm gibt es eine Besonderheit. Das „–"-Zeichen. Damit ist es möglich, auch rückwärts zu schreiten oder Drehungen im Uhrzeigersinn durchzuführen. Wenn eine Zahl, wie „–#90" übersetzt wird, so ergibt sich z. B. bei „21 –#90.W" die Übersetzung „21 A6 FF". Die Zahl FFA6 ist die sedezimale Zweierkomplementdarstellung der Zahl –90. Im Speicher wird diese Zahl in zwei Hälften zerlegt und verdreht herum abgespeichert, so daß sich „A6 FF" ergibt. Die verdrehte Reihenfolge ist nötig, damit sie der Prozessor Z80 versteht. Das hat der Hersteller so bestimmt. Aber die Umrechnung und Anordnung übernimmt zum Glück das Grundprogramm, so daß man sich darum nicht kümmern muß.



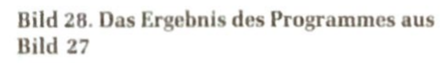


Bild 28. Das Ergebnis des Programmes aus Bild 27

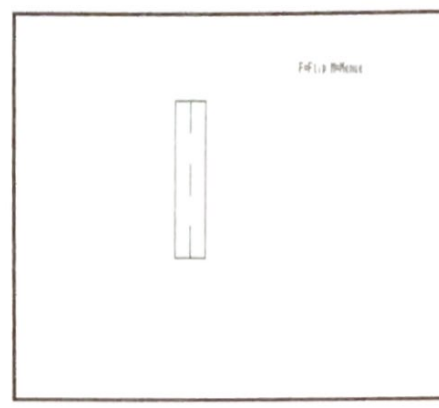


Bild 30. Die Straße mit Mittelstrich

Speicher ansehen

+=weiter -=rueckw R=Adr M=Menue

-- 0 1 2 3 4 5 6 7 8 9 A B C D E F

Bild 31. So sieht der Speicherauszug aus, wenn das Programm aus Bild 27 eingetippt ist

```
21 #90.W              21 #40.W
CD DREHE              CD SCHREITE
21 #20.W              CD HEBE
CD SCHREITE           21 #40.W
21 -#90.W             CD SCHREITE
CD DREHE              CD SENKE
21 #200.W             21 #40.W
CD SCHREITE           CD SCHREITE
21 -#90.W             21 #90.W
CD DREHE              CD DREHE
21 #20.W              21 #20.W
CD SCHREITE           CD SCHREITE
21 -#90.W             21 #90.W
CD DREHE              CD DREHE
21 #40.W              21 #200.W
CD SCHREITE           CD SCHREITE
CD HEBE               21 #90.W
21 #40.W              CD DREHE
CD SCHREITE           21 #20.W
CD SENKE              CD SCHREITE
                      C9
```

Bild 29. Das Programm malt eine Straße

Eine Straße wird gezeichnet

Nun ein weiteres Beispiel. Eine kleine Straße soll gezeichnet werden. Dazu das Programm in *Bild 29*. Wenn man das Programm eingibt und startet, so ergibt sich *Bild 30*. In *Bild 31* ist der Speicherinhalt einmal vollständig zum Vergleich abgebildet. Dabei ist die Adresse 8879 mit dem Inhalt „C9" die letzte definierte Speicherzelle. In Ihrem Computer können danach andere Werte stehen, das stört aber nicht weiter.
Unter jedem sedezimalen Wert steht ein ASCII-Zeichen. Dies ist wie schon gesagt die Interpretation des Grundprogramms, wenn man den sedezimalen Code, der darüber steht, als ASCII-Code ansieht (siehe Folge „Schreiben lernen") und auf den Bereich 0 bis 7F abbildet. Dabei wird das Bit 7 ignoriert, dann zum Beispiel ist CD kein ASCII-Element. CD ist dual 11001101. Wenn man aber die erste Stelle wegläßt ergibt sich 100 1101. Dies ist sedezimal dargstellt 4D und 4D entspricht in ASCII dem Zeichen „M". Diese Darstellung dient nur der zusätzlichen Information. Sie wird erst später gebraucht.

Übrigens besitzt der Bildschirm eine Auflösung von 512 × 256 Punkten. Für die Zeichensprache wird der Bereich auf 512 × 512 Punkte erweitert, um symmetrische Darstellungen zu ermöglichen. Die Schildkröte beginnt bei Start des Programms immer in der Mitte des Bildschirms zu zeichnen. Sie zeigt dabei in Richtung oberer Bildrand.

Aufgaben

1. Es soll ein Dreieck gezeichnet werden. Wie sieht das Programm aus?

2. Man versuche eine einfache Haus-Darstellung auf den Bildschirm zu bringen.

3. Was geschieht, wenn man die Befehle „21 #30.W" und „CD DREHE" vor das Quadratprogramm stellt? Also Programm nochmals eingeben und dabei mit der neuen Sequenz beginnen und dann das restliche Programm eingeben.

4. Ein Sechseck soll gezeichnet werden.

5. Was passiert, wenn man zu lange Linien zeichnet? Dazu versuche man folgendes Programm „21 #400.W", „CD SCHREITE", „21 #170.W", CD DREHE", „21 #400.W" und „CD SCHREITE" sowie „C9".

Rolf-Dieter Klein

Blumen mit Schleife

Mikroelektronik, Folge 13

Der Sinn des letzten Kapitels war, daß Sie feststellen sollten, daß mit ganz wenig Befehlen, den Grafikbefehlen unserer kleinen Grafiksprache, schon sehr sinnvolle Programme geschrieben werden konnten. In diesem Kapitel werden nun Befehle geschildert, die es erlauben, einmal durchprogrammierte Programmteile immer wieder zu benutzen. Das gibt der Sprache eine kräftige Struktur mit der man leicht umgehen kann und die es erlaubt, auch mächtige Programme sicher zu beherrschen.

Es soll ein Kreis auf dem Bildschirm gezeichnet werden. Das geht mit unseren Mitteln sehr einfach. Die Schildkröte kann als kleinsten Schritt um einen Bildpunkt fortschreiten und der kleinste Winkel, um den sie sich drehen kann, beträgt 1 Grad.
Das hat seinen guten Grund. Denn ein Bild auf dem Bildschirm besteht immer aus einzelnen Punkten. Damit kann man zwar keinen exakten Kreis darstellen, die Kreisform läßt sich aber ganz gut annähern.
Was würde geschehen, wenn man die Schildkröte um einen Schritt schreiten ließe und dann um ein Grad drehen würde, dann wieder einen Schritt schreiten, und dann um ein Grad drehen usw...?

```
21 #360.W
CD SCHLEIFE
21 1.W
CD SCHREITE
21 1.W
CD DREHE
CD ENDSCHLEIFE
C9
```

Bild 1. Dies Programm zeichnet einen Kreis, probieren Sie es aus

Es entstünde ein Art Kreis. Wann würde die Schildkröte wieder den ursprünglichen Ort erreichen? Wenn sie einen Winkel von 360 Grad durchlaufen hätte, denn dann hätte sie sich einmal um sich selbst gedreht und würde wieder in die Anfangsrichtung zeigen.
Es würde also ein 360-Eck gezeichnet worden sein.
Wenn ein solches Programm eingegeben werden soll, so hat das einen Haken, denn die Sequenz

21 #1.W
CD SCHREITE
21 #1.W
CD DREHE

muß offenbar 360mal eingegeben werden. Das sind über tausend Befehle. Dabei müssen immer wieder die gleichen Elemente niedergeschrieben werden.
Bei richtigen Computern gibt es dazu ein Hilfsmittel, die Schleife (auch LOOP genannt), die ständige Wiederholungen ein und desselben Programmstücks unterstützt, ohne daß man das Stück immer wieder niederschreiben muß. Auch in unserem Grundprogramm gibt es eine derartige Hilfe. Die beiden Befehle:

CD SCHLEIFE

und

CD ENDSCHLEIFE

Mit Schleifen gezielt wiederholen

Mit „CD SCHLEIFE" wird der Beginn einer Wiederholung markiert. Mit „CD ENDSCHLEIFE" wird das Ende markiert. Alle zwischen diesen beiden Befehlen stehenden Anweisungen werden wiederholt. Wie oft? Dazu muß ein Lade-Befehl vorangestellt werden, dessen Lade-Wert die Anzahl der Durchläufe angibt. Das vollständige Programm ist in *Bild 1* abgedruckt. Das Programm kann auch als Diagramm dargestellt werden, man nennt die Darstellungsform in *Bild 2* Struktogramm.
Nun zur Eingabe des Programms. Es wird ab Adresse 8800 eingegeben. Wichtig! Nicht das „C9" am Schluß vergessen. *Bild 3* zeigt das Ergebnis des Programms auf dem Bildschirm.

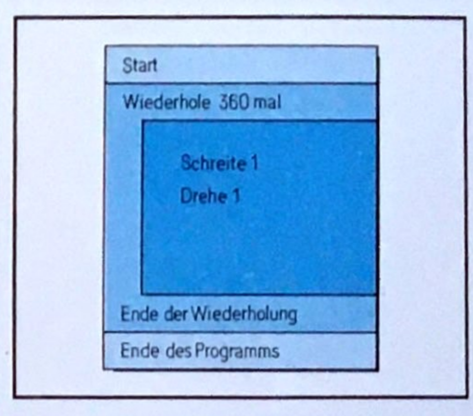


Bild 2. Das Programm aus Bild 1 als Struktogramm. Das bedeutet, daß in den Rechtecken Handlungen des Computers mit Worten niedergeschrieben sind und zwar in der Reihenfolge, in der der Computer sie ausführen soll. Besteht so eine Handlung aus mehreren Einzelhandlungen, dann kann man diese in Unter-Kästen notieren

Aufgaben

1. Was passiert, wenn man in der Schleife um 2 schreitet?
2. Was passiert, wenn man die Schildkröte in der Schleife um je 2 Grad dreht?
3. Wie kann man kleinere Kreise zeichnen?

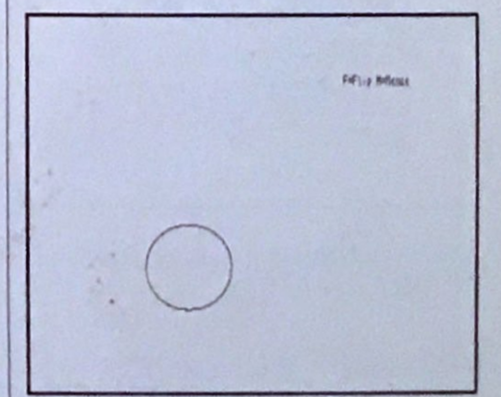


Bild 3. Das Ergebnis eines Programmlaufes

Eine Blume aus Teilen zusammensetzen

Es soll eine Blume gezeichnet werden. Diese Blume soll aus Blüten und einem Stengel bestehen. Das Kreisprogramm kann man dazu hervorragend verwenden. Zunächst soll ein Blütenblatt gezeichnet werden. Dieses Blütenblatt setzen wir aus zwei Viertelkreisen zusammen. *Bild 4* zeigt ein Schema.
Also gilt es, zuerst einen Viertelkreis zeichnen zu können. Wie das geht, müßte aber klar sein. Man zeichnet einfach

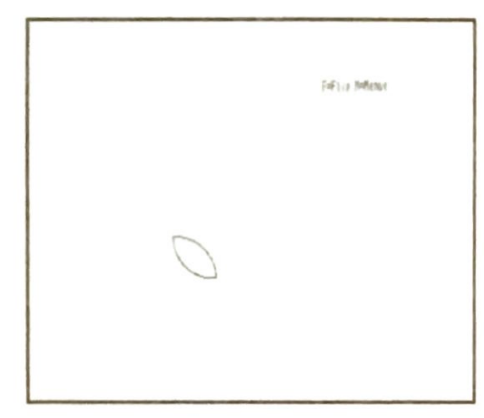


Bild 4. Ein Blütenblatt, hier aus zwei Viertelkreisen zusammengesetzt

den vierten Teil eines Kreises, verwendet also nur 90 Wiederholungen anstelle der 360.
Bild 5 zeigt das Viertelkreisprogramm. Dort ist noch eine Besonderheit zu sehen, nämlich die erste Zeile. Wenn man diese mit eintippt, so bekommt das Programm einen Namen und man kann spä-

```
VIERTELKREIS:=$
21 #90.W
CD SCHLEIFE
21 1.W
CD SCHREITE
21 1.W
CD DREHE
CD ENDSCHLEIFE
C9
```

Bild 5. So zeichnet Ihr Computer einen Viertelkreis

ter einfach den Namen anstelle der Adresse verwenden. Die Eingabe des Namens zeigt *Bild 6*. Wenn man anschließend die Taste „CR" drückt, verschwin-

```
aendern

Adr:8800

8800 : 00
8801 : 00

8800 viertelkreis:=$

+ =Adr + 1   - =Adr - 1
M =Menue     R =Adr
cr =ein Befehl weiter
```

Bild 6. Mit dieser Anweisung wird ein Name mit dem Computer vereinbart. Er weiß danach, daß die Adresse 8800 jetzt Viertelkreis heißt

det die Eingabe wieder, aber das Grundprogramm hat sich den Namen gemerkt. Nun kann das restliche Programm eingegeben werden. Wenn man den vorherigen Versuch durchgeführt hat, so braucht man nur die Zeile „21 #90.W" neu eingeben, der Rest ist ja identisch mit dem Kreisprogramm.
Nun geht man zurück ins Grundprogramm und ruft den Befehl „2" auf, also „Starten des Programms". Dann erscheint die Meldung „ADR.:" und jetzt kann man den Namen „VIERTELKREIS"

```
BLATT:=$
CD VIERTELKREIS
21 #90.W
CD DREHE
CD VIERTELKREIS
21 #90.W
CD DREHE
C9
```

Bild 7. Dieses Programm benutzt Viertelkreis als Unterprogramm. Viertelkreis ist damit in den Befehlssatz der Grafik-Sprache aufgenommen

eingeben. Nach dem „CR" erscheint der Viertelkreis auf dem Bildschirm.

Die Unterprogrammtechnik: Ein mächtiges Werkzeug

Wir wollten ein Blatt zeichnen und nicht nur einen Viertelkreis. Das Programm muß also noch erweitert werden. Dazu verwenden wir die sogenannte Unterprogrammtechnik. In unserem System können wir das Programm „Viertelkreis" als neues Sprachelement verwenden und rufen es einfach mit dem Befehl „CD" auf.
Für ein Blatt benötigt man zwei Viertelkreise, die aneinander gesetzt werden müssen. *Bild 7* zeigt die Lösung. Dabei wird zunächst ein neuer Name definiert, nämlich „BLATT". Unter dieser Bezeichnung läßt sich das Programm später aufrufen. Man muß sich also keine Adresse merken. Dann wird mit „CD VIERTELKREIS" unser vorhergehendes Programm aufgerufen, so als ob VIERTELKREIS ein eigener Befehl sei. Danach muß die Schildkröte um 90 Grad gedreht werden, ehe der zweite Viertelkreis angeschlossen werden kann. Und dann wird erneut das Unterprogramm „VIERTELKREIS" aufgerufen. Zum Schluß wird nochmals um 90 Grad gedreht. Dies geschieht, damit die Schildkröte wieder in die Ausgangsrichtung blickt, denn sie hat dann insgesamt eine Drehung von 360 Grad durchlaufen. Damit wird das weitere Arbeiten einfacher. Man sollte bei eigenen Programmen immer darauf achten, bei geschlossenen Figuren die Schildkröte auch insgesamt um 360 Grad oder ein Vielfaches davon zu drehen.

Die Blume wird eingegeben

Das alte Programm „VIERTELKREIS" darf natürlich nicht zerstört werden und daher gehe man wie folgt vor:
1. Änderungsmenü aufrufen.
2. Eingabe der Adresse 8800 oder des Namens „VIERTELKREIS" (große und kleine Buchstaben sind zugelassen).
3. Das alte Programm wird jetzt sichtbar. Nun muß man die Taste „CR" so oft drücken, bis der letzte Befehl des schon geschriebenen Programms sichtbar wird, also der Befehl „C9". Dann drücke man noch zweimal die Taste „CR", bis der Befehl „C9" ganz oben im Bild steht. Damit hat man einen Speicherplatz angewählt, der noch nicht durch ein Programm belegt ist. Dort kann man jetzt das neue Programm eingeben. Es beginnt mit der Definition des Namens, also mit „BLATT:=$" und sieht dann wie *Bild 8* aus. Danach drückt man die Taste „CR" und die Eingabe verschwindet wieder, das Grundprogramm hat die Eingabe angenommen. Achtung! Am restlichen Bild ändert sich bei Namensdefinitionen nichts. (Das Zeichen „$" ist übrigens die Abkürzung für die links neben dem Ka-


```
aendern

Adr:viertelkreis

8815 : C9
8816 : FF
8817 : 00

8816 blatt:=$

+ =Adr + 1   - =Adr - 1
M =Menue     R =Adr
cr =ein Befehl weiter
```

Bild 8. Der Name von Blatt wird vereinbart

sten stehende Adresse, man könnte auch schreiben: „BLATT:=8816", das aber nur in diesem Fall, denn der Wert der Adresse ist von der jeweiligen Lage des Programms abhängig.) Die Eingabe mit dem Dollar-Zeichen ist sehr bequem.
Als nächstes folgt also „CD VIERTELKREIS", dann „21 #90.W" usw. *Bild 9* zeigt einen Ausschnitt.
Wenn das Programm gestartet wird, und diesmal mit dem Namen „BLATT", so ergibt sich wirklich ein Blatt. Hier zeigt sich der Vorteil der Namensgebung. Wer hätte noch gewußt, daß die Startadresse des Programmes Blatt 8816 ist? Die Eingabe des Namens „BLATT" ist da viel einfacher.

```
aendern

Adr:viertelkreis

8816 : CD 00 88 ; CD VIERTELKREIS
8819 : 00
881A : 00

8819 21 #90.W

+ =Adr + 1   - =Adr - 1
M =Menue     R =Adr
cr =ein Befehl weiter
```

Bild 9. Sie können jetzt sicher schon selbständig weiterprogrammieren

Nun werden die Blätter zu einer Blüte zusammengesetzt. Dazu wird die Form Blatt einfach 5mal ausgegeben, wobei jedesmal eine Drehung von 72 Grad durchgeführt wird. Da 5 · 72 = 360 ist, ergibt sich eine geschlossene Figur.
Bild 10 zeigt das entsprechende Programm und *Bild 11* das Ergebnis bei Aufruf des Programms „BLUETE". Zur Programmeingabe verfährt man wie im vorherigen Beispiel. Zuerst wird das Ende des bisher eingegebenen Programmsystems gesendet. Dabei kann man bei der Adresse „BLATT" anfangen zu suchen bis „C9" im Bild oben erscheint. Dann kann das neue Programm eingegeben werden.

```
BLUETE:=$
21 #5.W
CD SCHLEIFE
CD BLATT
21 #72.W
CD DREHE
CD ENDSCHLEIFE
C9
```

Bild 10. Dies Programm malt eine Blüte, wenn es gleichzeitig mit Viertelkreis und Blatt im Speicher steht. Es benutzt nämlich diese beiden Unterprogramme

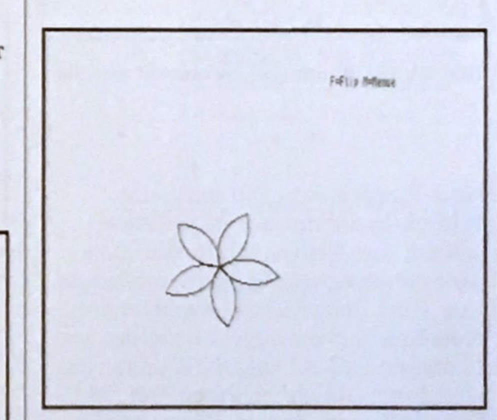


Bild 11. Das ist die Blüte

```
BLUME:=$
CD BLUETE
21 -#150.W
CD SCHREITE
CD BLATT
21 -#10.W
CD SCHREITE
21 -#90.W
CD DREHE
CD BLATT
21 #90.W
CD DREHE
21 -#40.W
CD SCHREITE
C9
```

Bild 12. Das Programm Blume

Wenn jetzt noch ein Stengel und zwei Blätter angefügt werden, ist die Blume fertig. *Bild 12* zeigt das Programm „BLUME" und *Bild 13* das Ergebnis auf dem Bildschirm.
In *Bild 14* ist der Speicherauszug abgedruckt. Damit kann man die Eingaben überprüfen.

Ein neuer Menüpunkt

Bisher hatten wir den Menüpunkt 4 = Symbole noch nicht verwendet. Das soll

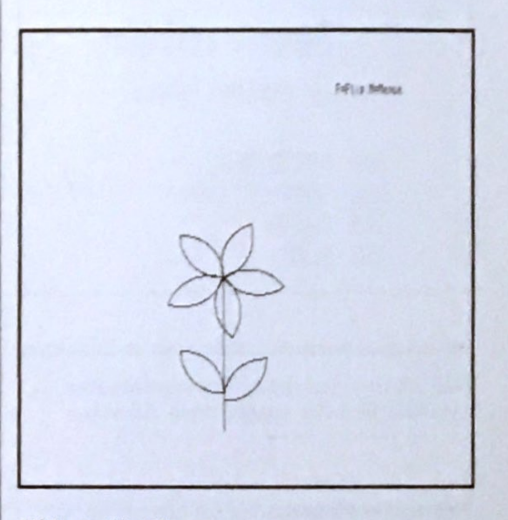


Bild 13. Die Blume

sich jetzt ändern. Wenn es aufgerufen wird, erscheint *Bild 15*. Dort sind alle definierten Namen eingetragen. Dabei steht links neben dem Namen der Wert, dem er zugeordnet wurde. Namen sind also bei uns nur eine andere Schreibweise für Zahlenwerte.

```
Speicher ansehen

+=weiter -=rueckw R=Adr M=Menue

--   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
8800 21 5A 00 CD 0F 00 21 01 00 CD 03 00 21 01 00 CD
8810 06 00 CD 12 00 C9 CD 00 00 21 5A 00 CD 06 00 CD
8820 00 88 21 5A 00 CD 06 00 C9 21 05 00 CD 0F 00 CD
8830 16 88 21 48 00 CD 06 00 CD 12 00 C9 CD 29 88 21
8840 6A FF CD 03 00 CD 16 88 21 F6 FF CD 03 00 21 A6
8850 FF CD 06 00 CD 16 88 21 5A 00 CD 06 00 21 D8 FF
8860 CD 03 00 C9 00 00 00 00 00 00 00 00 00 00 00 00
8870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Bild 14. Der Speicherauszug von Blume

Und so arbeitet auch die Zuordnung der Namen „SCHREITE", „DREHE". Auch diese Namen stehen nur anstelle von Zahlenwerten. Man könnte ebensogut die entsprechenden Zahlenwerte eingeben. Doch Namen haben hier einen Vorteil. Das Grundprogramm kann sie überprüfen. Wird ein Name eingegeben, der noch nicht definiert war, so erkennt es das Grundprogramm. Bei einer Befehlseingabe im Änderungsmenü bleibt die

```
Symbole ausgeben

+=weiter N=Nochmal M=Menue

8800  VIERTELKREIS
8816  BLATT
8829  BLUETE
883C  BLUME
```

Bild 15. Das sind die bisher vereinbarten Symbole und die zugehörigen Adressen

Adresse stehen und zeigt einem damit, daß ein Fehler gemacht wurde. Wenn man einen nicht definierten Namen als Adresse angibt, so wird z. B. beim Startmenü gar kein Programm ausgeführt, es meldet sich gleich das Grundprogramm.

Wenn man andererseits eine falsche Zahl als Adresse irrtümlich eingibt, so kann das Grundprogramm dies nicht kontrollieren – und das Programm „hängt sich" unter Umständen auf. Dann meldet sich das Grundprogramm nie wieder. Das eingegebene Programm wird dann sogar manchmal zerstört. Also: Namen verwenden spart Arbeit.

```
1 = IO lesen
2 = IO setzen
3 = Einzelschritt
W = weiter
```

Bild 16. Dieses Menü enthält den Punkt Einzelschritt

Fehler, Fehler!

Fehlersuche ist das wichtigste beim Programmieren, denn zunächst sind in einem längeren Programm immer irgendwelche Fehler, die man finden und beseitigen muß. Dazu benötigt man den Menüpunkt „Einzelschritt". Um dort hinzugelangen, betätigt man die Taste „W" im Grundmenü und dann die Taste „CR", und nochmals „W" und „CR". Dann erscheint *Bild 16*. Man wählt „Einzelschritt" durch die Taste „3" gefolgt von einem „CR".
Hier wird nun genau wie beim Startmenü die Adresse angefragt. Man gibt z. B. „BLUME" ein. *Bild 17* zeigt die Eingabe.



Bild 17. Die Blume soll untersucht werden

Nach Eingabe von „CR" erscheint *Bild 18*. In der unteren Bildhälfte erscheint eine Fülle von Informationen, die aber zunächst nicht alle von Bedeutung sind. Rechts unten erscheint der erste Befehl, der ausgeführt werden soll. In diesem Fall „CD BLUETE", denn das ist der erste Befehl im Programm „BLUME". Will man, daß der Computer diesen Befehl ausführt, so drücke man die



Bild 18. Vor dem ersten Schritt

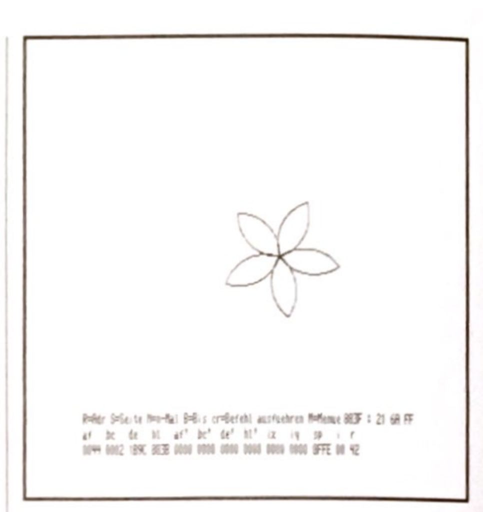
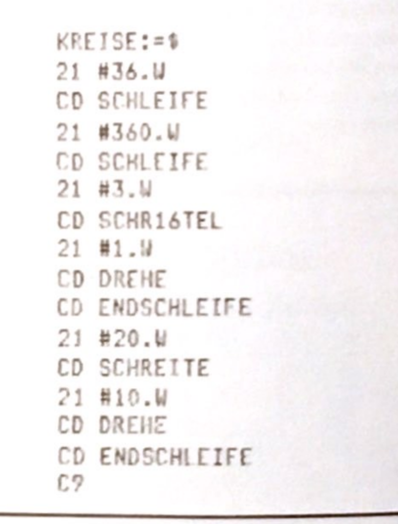


Bild 19. Und danach

Taste „CR". Es ergibt sich *Bild 19*. Der nächste Befehl, der ausgeführt wird ist „21 6A FF", also ein Lade-Befehl. Wenn man nun „CR" drückt, wird er ausgeführt, und es erscheint der nächste Befehl.
So kann man Befehl für Befehl schrittweise ausführen und damit Fehlern auf die Spur kommen.



```
KREISE:=$
21 #36.W
CD SCHLEIFE
21 #360.W
CD SCHLEIFE
21 #3.W
CD SCHR16TEL
21 #1.W
CD DREHE
CD ENDSCHLEIFE
21 #20.W
CD SCHREITE
21 #10.W
CD DREHE
CD ENDSCHLEIFE
C9
```

Bild 20. Das Programm Kreise mit neuem Befehl und verschachtelten Schleifen

Der Befehl „CD BLUETE" besteht selbst wieder aus einzelnen Befehlen. Diese werden aber nicht schrittweise durchlaufen, da es einen Namen dafür gibt. Will man „BLUETE" testen, so muß man „BLUETE" selbst als Startadresse anwählen. Hat man einen Fehler gefunden, so muß man die entsprechende Stelle mit dem Änderungsmenü verändern und das Programm von da an ggf. neu eingeben.

mikrocomputer schritt für schritt

Bei Programmen, die man selbst entwikkelt, empfiehlt es sich daher, sie erstens sofort Unterprogramm für Unterprogramm auszutesten und zweitens Lükken zwischen den Programmen zu lassen, um dort später weitere Befehle unterbringen zu können, falls man welche vergessen haben sollte. Man kann dazu zum Beispiel den NOP-Befehl verwenden, dessen Code 00 lautet. Er wird einfach zwischen Programmelementen passend eingestreut. Beispiel:

```
21 #90.W
CD DREHE
00 00 00 00 00
21 #1.W
CD SCHREITE
```

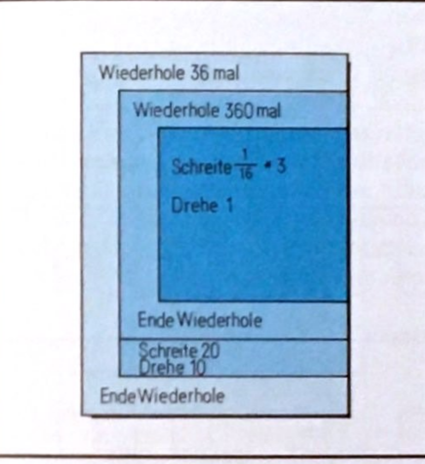


Bild 21. Das Struktogramm zu Kreise

Ein neuer Befehl,

CD SCHR16TEL, damit ist es möglich um einen ¹⁄₁₆-Punkt vorwärts zu schreiten. Diesen Befehl benötigt man zum Beispiel, wenn man irgendwelche Kreisradien erzeugen will.

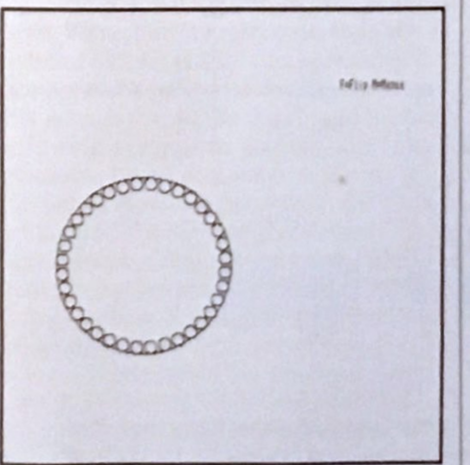
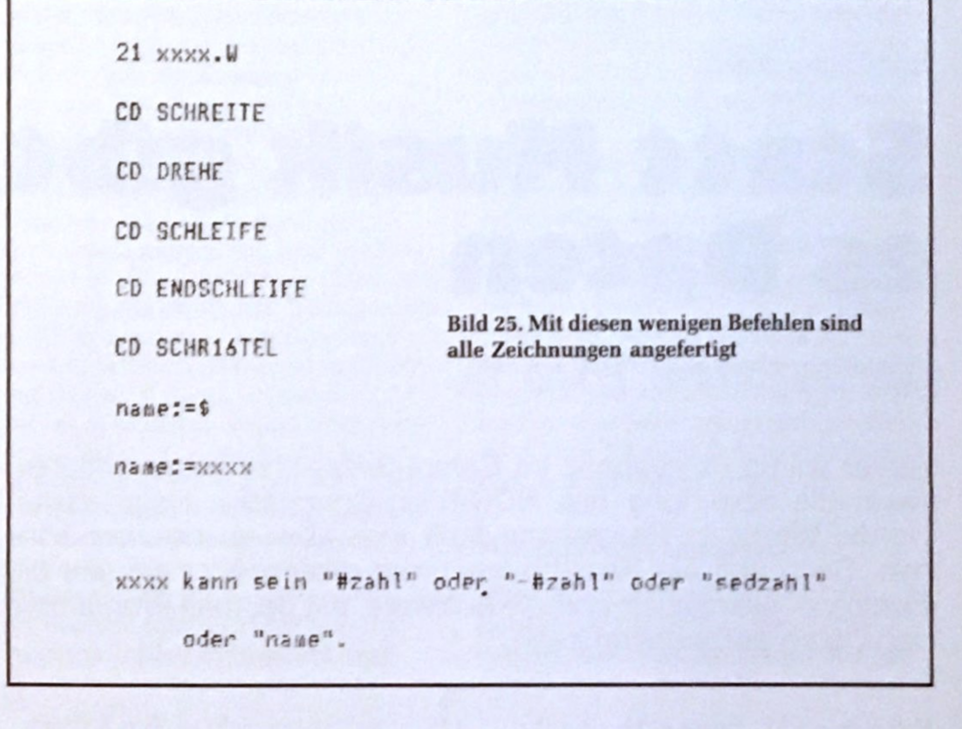


Bild 22. Das sind 36 Kreise in einem 36-Eck



```
21 xxxx.W

CD SCHREITE

CD DREHE

CD SCHLEIFE

CD ENDSCHLEIFE

CD SCHR16TEL

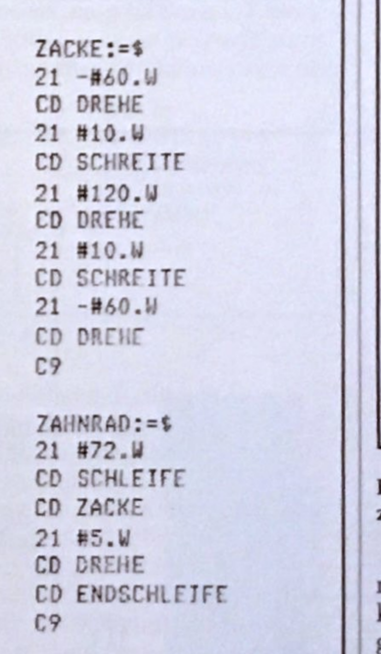
name:=$

name:=xxxx


xxxx kann sein "#zahl" oder "-#zahl" oder "sedzahl"

     oder "name".
```

Bild 25. Mit diesen wenigen Befehlen sind alle Zeichnungen angefertigt



```
ZACKE:=$
21 -#60.W
CD DREHE
21 #10.W
CD SCHREITE
21 #120.W
CD DREHE
21 #10.W
CD SCHREITE
21 -#60.W
CD DREHE
C9

ZAHNRAD:=$
21 #72.W
CD SCHLEIFE
CD ZACKE
21 #5.W
CD DREHE
CD ENDSCHLEIFE
C9
```

Bild 23. Zahnrad benutzt Zacke

Ein Beispiel zeigt *Bild 20*. Dabei ist im Programm noch eine Besonderheit untergebracht. Es sind diesmal zwei Schleifen-Befehle ineinandergeschachtelt. *Bild 21* zeigt das Struktogramm. Die innere Schleife ist das Kreisprogramm. Die äußere Schleife erzeugt ein 36-Eck. Beides ergibt zusammen die Figur in Bild 22.
Jetzt eine letzte Aufgabe. Es soll ein Zahnrad gezeichnet werden. Dazu kann man zuerst eine Zacke definieren, die

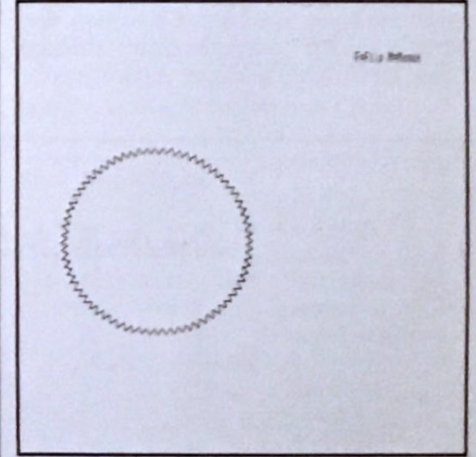


Bild 24. Ein Zahnrad vom Computer gezeichnet

man dann zu einem Zahnrad ausbauen kann. *Bild 23* zeigt das vollständige Programm und *Bild 24* das Ergebnis auf dem Bildschirm.
Bild 25 zeigt die Liste von Befehlen, die bis hierher verwendet wurden.

Aufgaben

1. Was passiert, wenn man ein „CD SCHLEIFE" oder „CD ENDSCHLEIFE"-Befehl vergißt?
2. Wie kann man das Quadrat-Programm jetzt kürzer schreiben?
3. Verschiedene Figuren sollen zunächst auf Papier gezeichnet werden und dann ein Programm dazu entwickelt werden. Beispiel: Dreiecke im Kreis angeordnet, Haus usw.

Rolf-Dieter Klein

Statt Musik gibt es Daten

Mikroelektronik, Folge 14

Bisher waren Programme im Computerspeicher immer verloren, wenn die Spannung des NDR-Klein-Computers ausgeschaltet wurde. Wertvolle Programme muß man aber abspeichern können. Dazu soll ein Kassettenrecorder dienen. Hier werden die Elektronik geschildert und die Software, mit der man Programme und Daten aufzeichnen kann.

Daß man auf Tonbändern Musik aufzeichnen kann, weiß jeder, daß man aber auch Daten für Computer darauf ablegen kann, ist nicht so bekannt. Allerdings kann man Daten nicht so ohne weiteres auf das Tonband bringen, sie müssen dazu aufbereitet werden. Dafür benötigt man eine spezielle Schaltung, ein Interface, die Kassetten-Baugruppe. Sie arbeitet folgendermaßen: Zunächst werden die Daten-Bytes, die im Computerspeicher liegen, durch das Grundprogramm nacheinander in einen Baustein (mit der Bezeichnung 6850) auf die CAS-Baugruppe befördert. Dieser Baustein zerlegt die angelieferten Bytes in die einzelnen Bits und überträgt diese einzeln hintereinander an die restliche Schaltung. Die Bits tragen die Information 0 oder 1, in Spannungswerten ausgedrückt: 0 V oder 5 V.

Man kann nun einmal versuchen, diesen Datenstrom direkt an den Tonbandeingang eines Kassettenrecorders zu führen. Dann wird man feststellen, daß bei einer Wiedergabe alles andere als das ursprüngliche Signal wieder herauskommen. So einfach geht es also nicht. Eine spezielle Schaltung muß das Datensignal so umwandeln, daß es ein Tonsignal wird. Erst dann wird es dem Kassettenrecorder zugeführt. Eine solche Schaltung nennt man Modulator. Umgekehrt muß aus einem solchen Signal auf Tonband das ursprüngliche Bitmuster wieder zurückgewonnen werden. Man sagt „es wird demoduliert". Anschließend werden die Bits wieder in einen Strom von Bytes verwandelt, die in den Speicher des Computers zurückgeladen werden können.

Man kann Daten nicht beliebig schnell an einen Kassettenrecorder übertragen, da ein Kassettenrecorder Schwingungen, die eine bestimmte maximale Frequenz überschreiten, nicht mehr aufzeichnen kann. Die Zahl der Bits, die pro Sekunde übertragen werden, ist eine wichtige Größe. Sie wird als Baudrate bezeichnet. Die CAS-Baugruppe, die gleich aufgebaut werden wird, ist für eine Übertragungsrate von 1200 Baud ausgelegt. Das bedeutet: 1200 Bit pro Sekunde können übertragen werden. Eine höhere Baudrate ist zwar möglich (bis zu 6000 Baud), jedoch benötigt man dann ausgesuchte Kassettenrecorder. *Bild 1* zeigt den Schaltplan der Kassettenschaltung, *Bild 2* den Bestückungsplan und *Tabelle 1* die Stückliste.

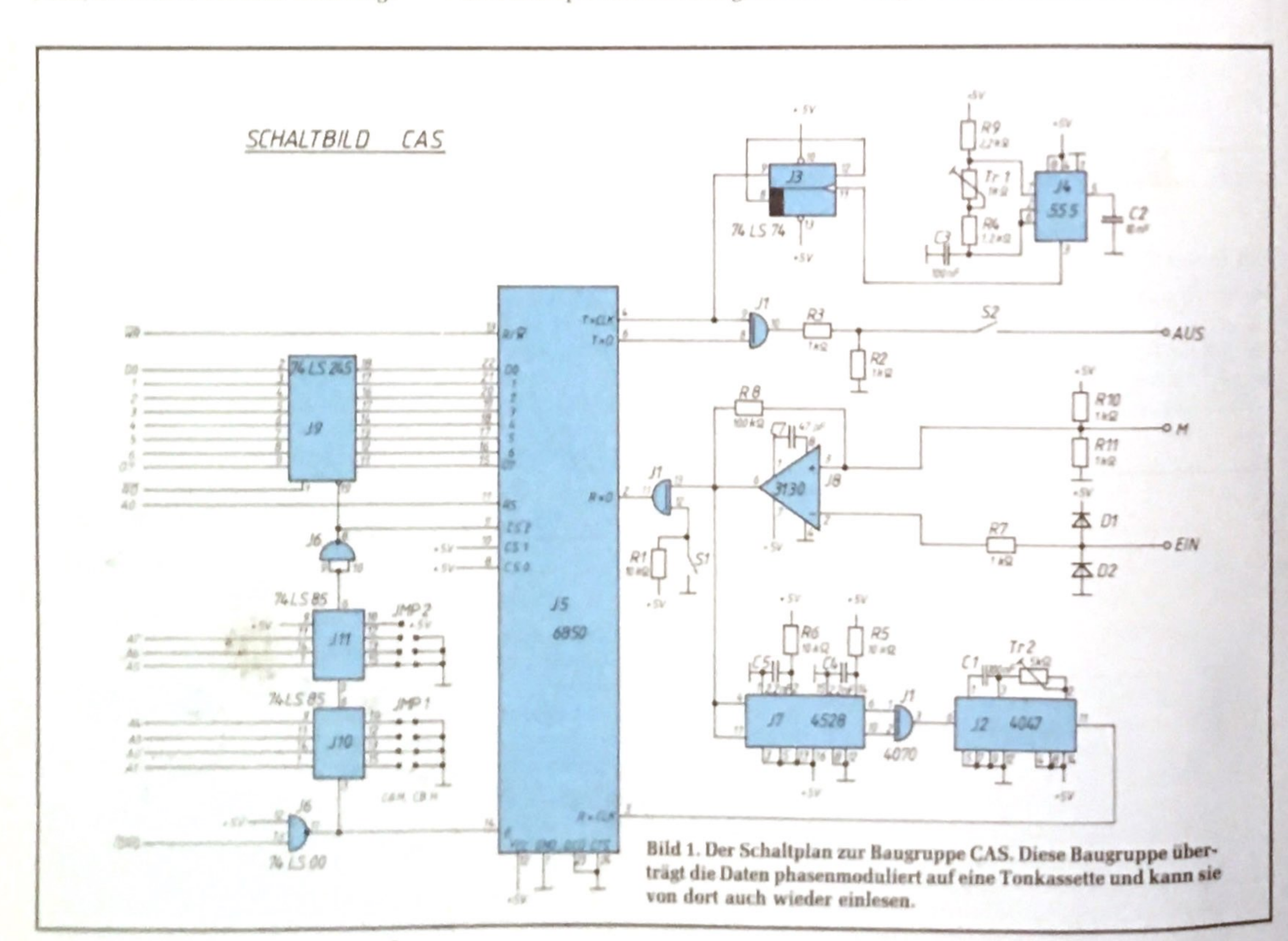


Bild 1. Der Schaltplan zur Baugruppe CAS. Diese Baugruppe überträgt die Daten phasenmoduliert auf eine Tonkassette und kann sie von dort auch wieder einlesen.

So funktioniert CAS

Es sei ganz grob geschildert, wie die Schaltung arbeitet. Der Baustein 6850 wird vom Computer über eine I/O-Adresse angesprochen, die mit dem Adreßdecoder, bestehend aus den beiden 7485-Bausteinen, eingestellt werden kann. Wenn der Computer ein Byte abgeliefert hat, sorgt eine spezielle Schaltung im 6850 dafür, daß dessen einzelne Bits zunächst vorn um eine 0 und hinten um zweimal 1 ergänzt werden. Ein Taktgenerator, bestehend aus dem Baustein 555 liefert einen präzisen Takt, der noch auf 1200 Hz halbiert wird und dem 6850-Baustein zugeführt wird. Im Baustein wird während einer Periode des Taktsignales nun immer je ein Bit des angelieferten und ergänzten Bytes genommen, beginnend bei der zugesetzten Null, dem Startbit, und dieses am Ausgang Txd ausgegeben. Präzise und synchron mit dem Taktsignal erscheint also am Ausgang die Folge der einzelnen Datenbits, vorweg eine Null, hinten daran zwei Einsen, die einem späteren Empfänger Zeit lassen sollen, das Empfangene Byte auch wieder auszuwerten.

Takt- und Datensignal sind beide an einen EXOR-Baustein geführt und werden dort gemischt. Dabei geschieht Entscheidendes (*Bild 1b*). Ein EXOR mit zwei Eingängen führt am Ausgang genau dann eine 1, wenn genau einer der Eingänge 1 ist. Dies führt dazu, daß bei einer Mischung von Daten- und Taktsignal am EXOR immer dann, wenn kein Signalwechsel bei dem Datensignal stattfindet und dieses auf 0 liegt, der Originaltakt anliegt. Wechselt zu Beginn einer Periode das Datensignal von 0 auf 1, so wird im gemischten Signal genau die Hälfte des Taktsignales zusätzlich gewartet, bis dort ein Polaritätswechsel von 0 auf 1 stattfindet. Dies geschieht analog, wenn das Datensignal von 1 auf 0 wechselt. Da der Baustein 6850 die Datenbits exakt synchron mit den Perioden der Taktschwingung abgibt, kann man, wenn man das Mischungsergebnis in der zweiten Hälfte einer jeden Takt-Periode anschaut, feststellen, ob von 0 auf 1 gewechselt wurde, denn dann ist dort der Wert 1 festzustellen, oder der Wert 0, wenn von 1 auf 0 gewechselt wurde.

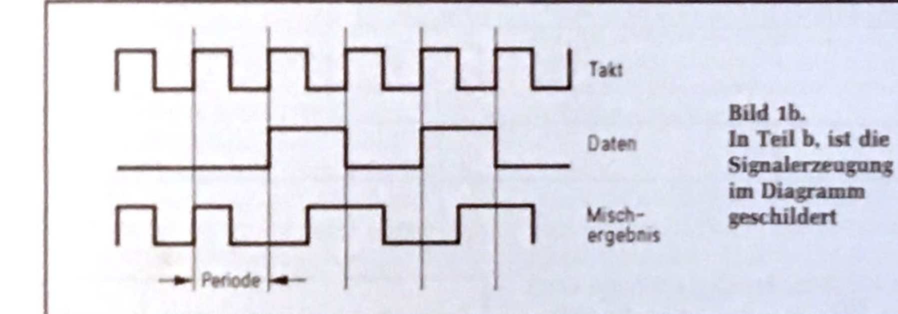


Bild 1b. In Teil b. ist die Signalerzeugung im Diagramm geschildert

Genau so etwa funktioniert die Auswerteschaltung, bei der zunächst das Signal verstärkt wird und dann im unteren Teil an Pin 11 des 4047 ein Signal erzeugt wird, das exakt die Mitte einer jeden Periode der Sendetaktschwingung aus dem modulierten Signal entnimmt. Das Monoflop MV3 (J2) ist so eingestellt, daß es genau ¾ einer Periode abwartet und dann einen Puls erzeugt, der den 6850-Baustein veranlaßt, den Wert des modulierten Signales an dieser Stelle am Eingang Rxd zu übernehmen. Das geschieht nämlich immer dann, wenn an Rx CLK eine steigende Flanke aus dem 4047-Baustein ankommt. Der Baustein CD 4528 erzeugt übrigens bei jedem Wechsel des Eingangssignales einen Puls und stellt so immer den Bezug zur Sendetakt-Periode fest. Der Schalter S1 hilft die richtige Polarität des Eingangssignales einstellen, denn manche Kassettenrecorder kehren die Polarität um.

Wenn Sie mit diesen Angaben versuchen, einen Impulsplan des Verfahrens aufzustellen, werden Sie noch Schwierigkeiten haben, denn es wurde nur das Prinzip geschildert. Es muß nämlich bei der Auswertung auf die richtige Signalflanke eingerastet werden und die Polarität des Wiedergabesignales muß invertiert sein. Vielleicht hilft das Buch „Mikrocomputer selbst gebaut und programmiert", das vertiefende Informationen zu allen Kapiteln bereit hält.

Wie man CAS justiert

In der Schaltung befinden sich zwei Schalter. S2 wird immer dann eingeschaltet, wenn eine Aufnahme durchge-

Tabelle 1. Die Stückliste zu CAS

Stück	Bezeichnung			
1	J1	4070	4 EXOR	
1	J2	4047	Monoflop	
1	J3	74 LS 74	D-Flipflop	
1	J4	555	Generator	
1	J5	6850	Serieller Baustein	
1	J6	74 LS 00	4 NAND	
1	J7	4528	Monoflop	
1	J8	3130	Operationsverstärker	
1	J9	74 LS 245	Bus-Transceiver	
2	J10, J11	74 LS 85	Vergleicher	
2	SO8	8polige IC-Fassung		
4	SO14	14polige IC-Fassung		
3	SO16	16polige IC-Fassung		
1	SO20	20polige IC-Fassung		
1	SO24	24polige IC-Fassung		
3	R1, R5, R6			10 kΩ
5	R2, R3, R7, R10, R11			1 kΩ
1	R8			100 kΩ
1	R4			1,2 kΩ
1	R9			2,2 kΩ
1	Tr1			1 kΩ
1	Tr2			5 kΩ
2	C1, 3			100 nF
1	C2			10 nF
2	C9, 6			100 nF
2	C4, 5			2,2 nF
1	C8			10 µF
1	C7			47 pF
2	D1, 2	Siliziumdioden		
1	St1	36polige Steckerleiste		
1	Platine mit Lötstopplack			
1	Dioden-Buchse			

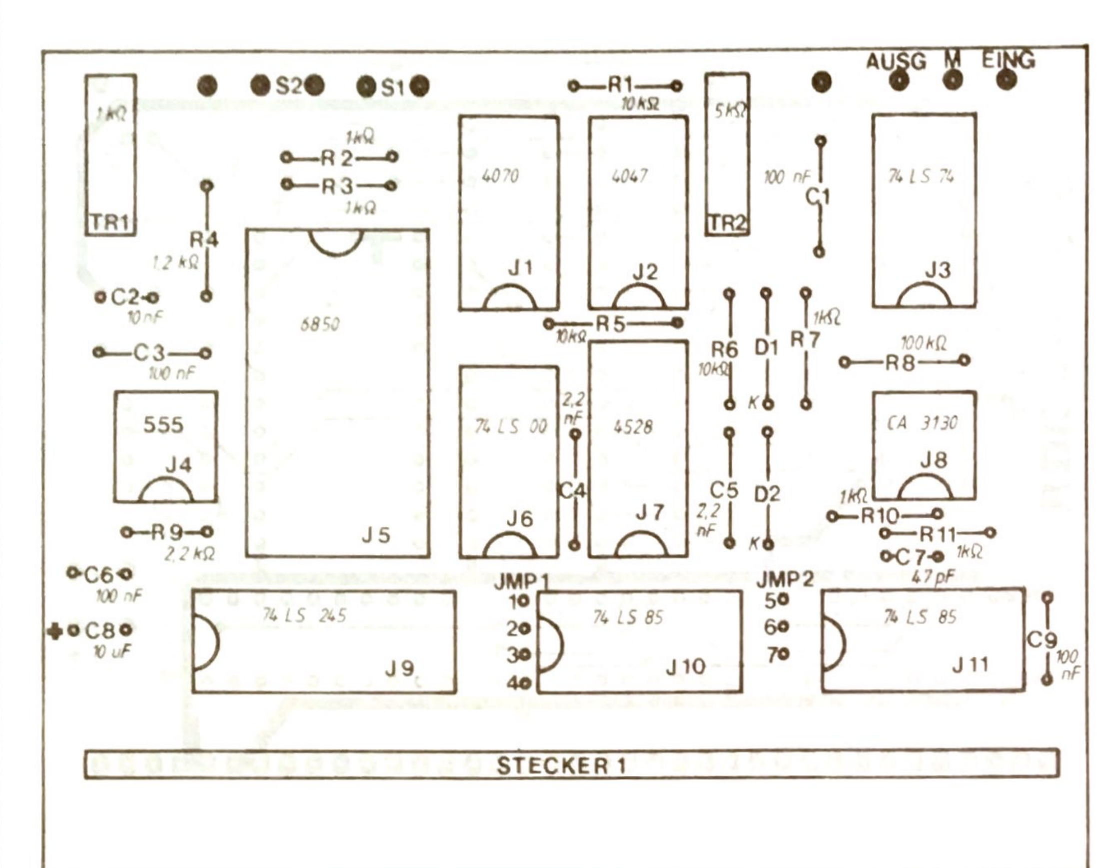


Bild 2. Der Bestückungsplan der Baugruppe CAS

führt werden soll. S2 leitet das Signal an den Kassettenrecorder. Bei manchen Kassettenrecordern wird das analoge Signal zwischen Aufnahme und Wiedergabe invertiert, bei anderen nicht. Um das Signal immer in die richtige Polarität zu bringen, gibt es den Schalter S1. Die richtige Lage muß man ausprobieren. Doch nun zum Abgleich.

Dazu wird ein Programm verwendet, das es ermöglicht, aus dem SBC2-Computer ein Oszilloskop zu machen. Das SKOP-EPROM wird dazu in Fassung 3 (IC9) auf der SBC2-Baugruppe anstelle des RAM-Bausteins eingesteckt. Dann wird der Computer eingeschaltet, es meldet sich wie bisher das Grundprogramm. Das Skop-Programm im EPROM 2716 muß jetzt gestartet werden. Dazu gehe man ins Startmenü und gebe die Adresse 8800 an. Dann meldet sich ein SKOP-Menü (*Bild 3*). Dort gibt es drei Menüpunkte. Der dritte liefert eine aktuelle Kurzerklärung, die wie in *Bild 4* aussieht.

Man benötigt noch eine IOE-Karte, um die Messungen durchführen zu können. Diese wird auf die Adresse 30 eingestellt (also Brücken 7 und 6 einlöten). Es wird das IC 74LS245, das ganz am Rand sitzt, benutzt. Die Bits 0 und 1 sind die Meßeingänge (siehe IOE-Schaltplan). *Bild 5* zeigt die Belegung an dem Benutzerstecker der IOE-Karte. Wenn man dort Stifte eingelötet hat, sollte man zwei Meßleitungen auf der Lötseite der IOE-Karte anbringen oder Buchsen verwenden.

```
RDK-Digital-Scop

1 = Periodendauer,      1-Kanal
2 = Vergleichsmessung, 2-Kanal
3 = Kurzerklaerung
```

Bild 3. Das Menü zum Skop-EPROM

```
*** Rev 1.0 ***
1. IOE-Karte auf Adresse 30h legen.
2. I/O 0-Port Bit 0 = Kanal 1.
3. I/O 0-Port Bit 1 = Kanal 2.
4. Kanal 1 ist auch Triggereingang.
5. Das Signal erscheint erst
   nachdem ein Signalwechsel
   am Triggereingang von 0 auf 1
   erfolgt.
6. Die Messungen erfolgen auf ca.
   5µs bis 6µs genau.
                         M=Menue
```

Bild 4. Die Kurzerklärung der Funktionen im EPROM

Der Computer hilft beim Abgleich

Nun zum Abgleich. Es muß zuerst der Sendetakt abgeglichen werden. Dazu gibt es das Trimm-Potentiometer TR1 auf der CAS-Baugruppe. Man muß die Meßleitung Kanal 1 mit dem Sendetakt verbinden. Der Sendetakt findet sich auf der Lötseite der CAS-Baugruppe an Pin

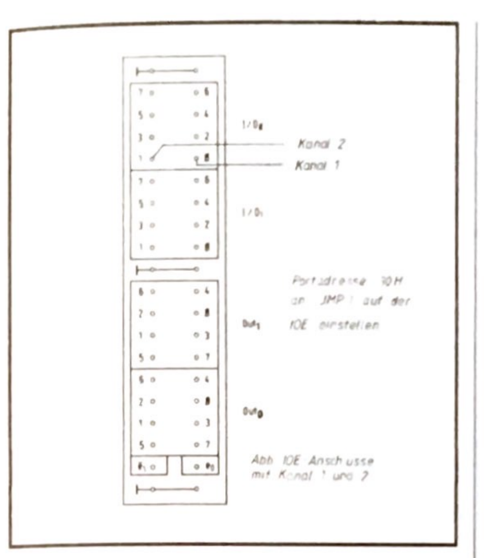


Bild 5. Mit diesen Anschlüssen auf der IOE-Platine muß CAS verbunden werden. Die IOE-Platine muß auf 30 adressierbar sein

4 des ICs 6850. Dort ist der kleine Buchstabe „c" aufgedruckt. Man lötet oder klemmt die Meßleitung dort fest. Nun schaltet man den Computer ein und startet bei Adresse 8800. Dann wählt man Menü 1 „Periodendauer 1-Kanal" aus. Jetzt muß auf dem Bildschirm ein Rechtecksignal sichtbar sein. Wenn nicht, so muß man alle Verbindungen überprüfen und auch die Brücken auf der IOE-Karte.

Man darf auch Kanal 1 nicht mit Kanal 2 verwechseln. Mit einem Schraubendreher wird dann TR1 solange verdreht, bis sich *Bild 6* einstellt. Die gemessene Periodendauer muß etwa 833 Mikrosekunden betragen. Aufgrund der Meßunsicherheit wird man aber nur eine Näherung einstellen können, was aber ausreicht.

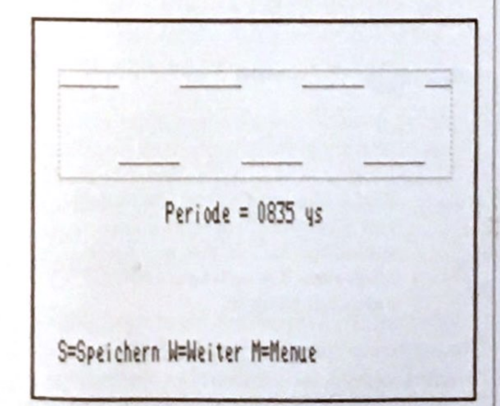


Bild 6. So sieht ein Rechtecksignal aus, mit dem Skop-EPROM auf dem Bildschirm dargestellt

Jetzt muß der Empfangsteil abgeglichen werden. Dazu muß man S2 auf Aufnahme stellen. Man benötigt ferner eine kleine Hilfsschaltung, um den Abgleich zu vereinfachen. *Bild 7* zeigt die Schaltung. Sie wird mit der CAS-Baugruppe verbunden.

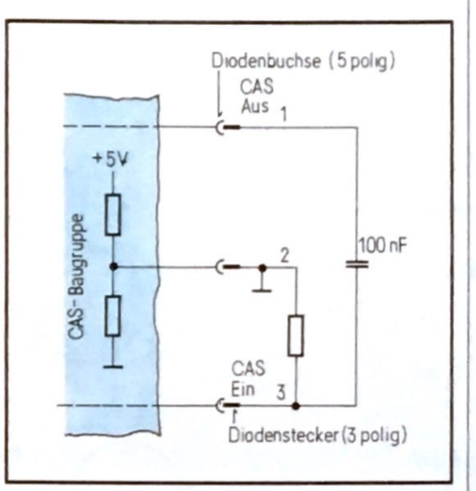


Bild 7. So sieht die Selbst-Test-Schaltung aus. Die Steckverbinder sollten auf der CAS-Seite eine DIN-Buchse (Diodenstecker) und auf der Schaltungsseite ein entsprechender dreipoliger Stecker sein

Nun wird der Kanal 1 mit dem Meßpunkt b, das ist Pin 3 des 6850 und Kanal 2 mit dem Meßpunkt a, das ist Pin 2 des 6850 (CAS-Baugruppe), verbun-

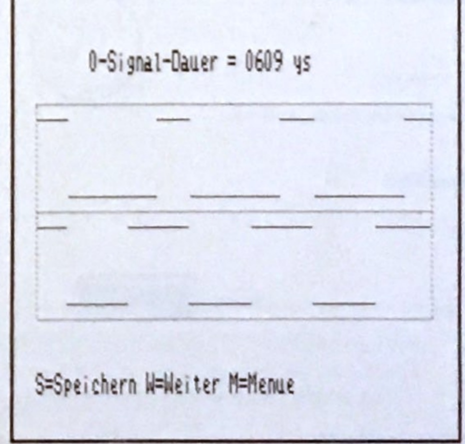


Bild 8. So etwa muß der Test auf dem Bildschirm aussehen

den. Man wählt den Menüpunkt 2, „Vergleichsmessung 2-Kanal", an und kann den Abgleich an TR2 durchführen. Dabei wird die 0-Signal-Dauer gemessen. Sie muß auf etwa 625 Mikrosekunden eingestellt werden. Der genaue Wert ist nicht so kritisch, die Einstellung mit dem besten Näherungsergebnis ist genau die richtige. *Bild 8* zeigt, wie es aussehen kann. Hier kann man auch die Funktion des Schalters S1 testen. Wenn man ihn umschaltet, so ändert sich die Polarität des unteren Signals.

Mit dem SKOP-EPROM kann man noch andere einfache Messungen durchführen, dabei ist die Frequenz auf etwa 190 kHz begrenzt, die minimale Frequenz auf etwa 370 Hz. Mit der Taste „S" kann man die Pulsform eines Signales speichern und in Ruhe betrachten. Die Triggerung, das ist die Auslösung des Meßvorgangs wird automatisch bei einer Signal-Änderung am Kanal ausgeführt. Ein Meßergebnis erscheint nur, wenn die Frequenz im zulässigen Bereich liegt.

Der Kassettenrecorder wird angeschlossen

Man verbindet CAS-AUS mit dem Eingang des Kassettenrecorders und CAS-EIN mit dem Ausgang des Recorders und CAS-Masse mit der Masseleitung des Kassettenrecorders. ACHTUNG! Die Leitung CAS-Masse hat einen Spannungspegel von etwa 2.5 V, es ist dies nicht die Masse des Computers (0V). Man darf sie nicht verwechseln, sonst funktioniert die Wiedergabe nicht.

Mit dem SKOP-EPROM kann man auch die Wiedergabe durch das Tonbandgerät kontrollieren. Man muß dazu eine Datenaufzeichnung durchführen.

Bild 9 zeigt die Belegung einiger gebräuchlicher Buchsen, entnommen den „Franzis Mini-Tabellen". Hier muß man etwas experimentieren, bis man seinen Kassettenrecorder richtig mit unserem Computer verbunden hat.

Beim Recorder sollte man darauf achten, daß er ein Bandlaufzählwerk und eine manuelle Aussteuerung besitzt. Manche Recorder arbeiten leider nicht so gut, da sie über Sprachfilter oder Entzerrer verfügen, die das CAS-Signal verfälschen. Es kommt nämlich auf dessen Phasenlage an. *Bild 10* zeigt eine Schaltung, die man verwenden kann, wenn es große Probleme gibt. Sie rundet das CAS-Signal ab, was von manchen Recordern besser verdaut werden kann. Ein Recorder, der alle unsere Forderungen erfüllt, ist der Typ MK 2000 von der Firma Waltham, München. Der Preis beträgt etwa 85 DM. Weitere Typen können von den Herstellern der Bausätze erfragt werden.

NF-Stecker und Buchsen

Mikrofonbuchse



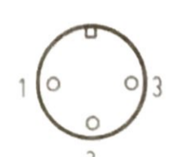
Mono
1 Aufnahme
2 Masse
3 Leer



Stereo L/R
1 Aufnahme, linker/rechter* Kanal
2 Masse
3 Leer
4 Aufnahme, rechter/linker* Kanal
5 Leer

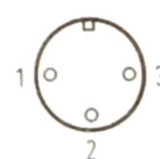
* Buchse für rechten Kanal bei getrennten Buchsen

TA-Buchse



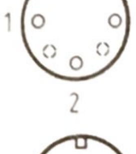
alt
1 Rechter Kanal
2 Masse
3 Linker Kanal u. Mono

TA/TB-Buchse



alt
1 Aufnahme TB
2 Masse
3 Wiedergabe TA/TB

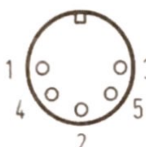
Radiobuchse oder Tuner



Mono
1 Aufnahme
2 Masse
3 Wiedergabe

Stereo
1 Aufnahme linker Kanal
2 Masse
3 Wiedergabe, linker Kanal
4 Aufnahme, rechter Kanal
5 Wiedergabe, rechter Kanal

neu
1 Leer
2 Masse
3 Linker Kanal
4 Leer
5 Rechter Kanal



neu
1 Aufnahme TB
2 Masse
3 Wiedergabe TA/TB
4 Leer
5 Mit 3 verbunden, falls Stereo TA angeschlossen ist (beide Kanäle parallel)

TB-Buchse Stereo



1 Aufnahme Mono und Aufnahme Stereo, linker Kanal
2 Masse
3 Wiedergabe Mono und Stereo, linker Kanal
4 Aufnahme Stereo, rechter Kanal
5 Wiedergabe Stereo, rechter Kanal

Lautsprecherbuchse



Masseanschluß

○ (zusätzlicher Anschluß = Umschalter)

Universalbuchse 7polig



1 Aufnahme
2 Masse
3 Wiedergabe
4 Leer oder mit 1 verbunden
5 Mit 3 verbunden
6, 7 Start/Stop-Fernbedienung mit Schaltmikrofon

Stereo-Kopfhörer



Stecker
1 Leer
2 Linke Masse
3 Rechte Masse
4 Linkes Signal
5 Rechtes Signal

Japanische und amerikanische Stecker

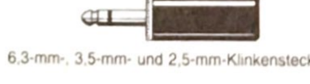
Aufnahme und Wiedergabe müssen entweder separat gesteckt werden oder es wird intern im Gerät umgeschaltet.

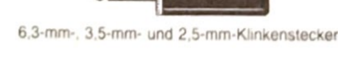


Chinch-Stecker

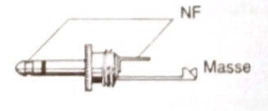








6,3-mm-, 3,5-mm- und 2,5-mm-Klinkenstecker





6,3-mm-Stereo-Klinkenstecker

Stecker für Spannungsversorgung







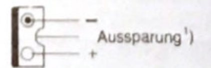


Bild 9. Das sind einige Varianten von Steckverbindern und Anschlußbeschaltungen, wie sie in modernen Kassettenrecordern auftreten können

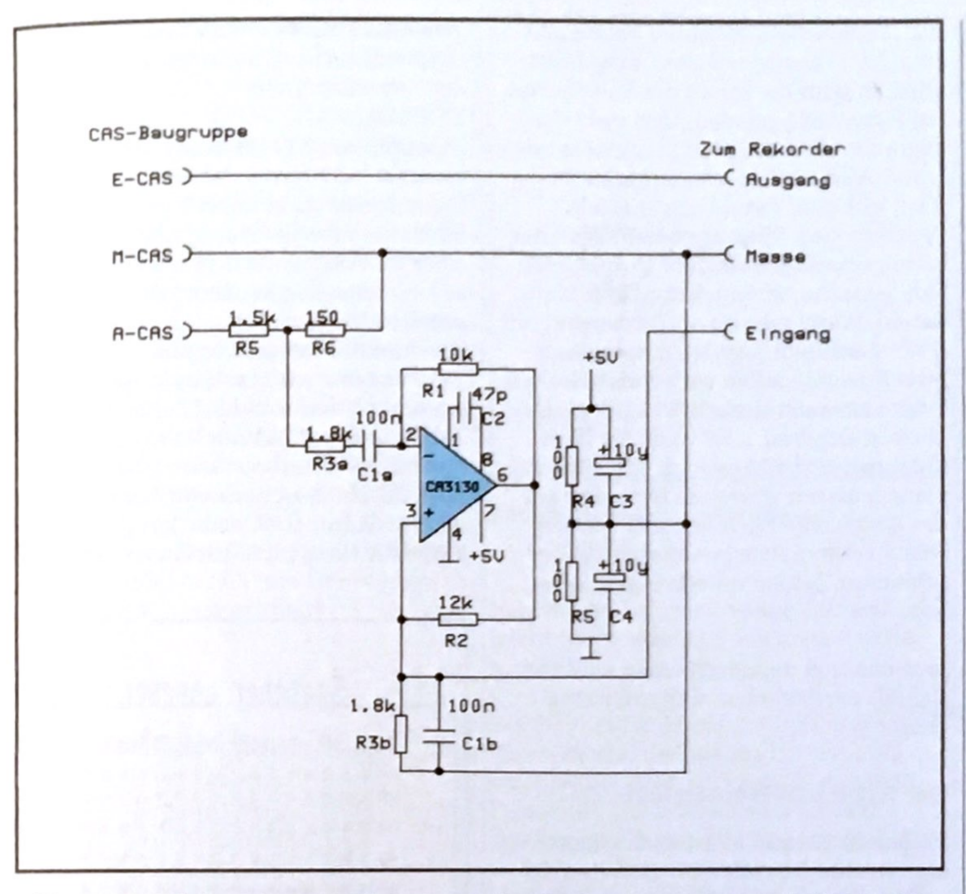


Bild 10. Diese Schaltung kann in schwierigen Fällen die Signale verbessern und die Aufzeichnung sicher machen

Die erste Datenaufzeichnung

1. Computer einschalten.
2. Mit „W" wird das Menü mit den Kassettenfunktionen angewählt.
3. Dann „2" für „2 = Speichern CAS" eingeben.
4. Es meldet sich das Kassettenmenü. *Bild 11* zeigt den Bildschirm.
5. Als erstes wird nach einer Startadresse gefragt. Zum Test die Adresse 8800 eingeben.

6. Als Endadresse wird 88FF eingeben, das soll die letzte Adresse sein, bis zu der abgespeichert werden soll.

7. Dann wird nach einem Namen gefragt. Jetzt muß man dem abzuspeichernden Inhalt einen Namen geben, der später, zur Kontrolle, wieder angezeigt werden wird. *Bild 12* zeigt den jetzigen Bildschirminhalt.

8. Ehe man nach der letzten Eingabe die Taste „CR" drückt, muß man den Kassettenrecorder auf Aufnahme schalten und starten. S2 muß geschlossen sein. Der Aussteuerungsanzeiger auf dem Kassettenrecorder sollte nun schon einen Pegel anzeigen.

```
auf CAS speichern

Startadr. :
Endadr.   :
Name      :
```

Bild 11. Startadresse, Endadresse und Namen des Datenpaketes werden vom Menüpunkt „auf CAS speichern" abgefragt

9. Jetzt die Taste „CR" drücken, die Aufzeichnung beginnt.
10. Nach ein paar Sekunden zeigt sich wieder das Grundmenü.
11. Nun den Recorder zurückspulen.
12. Den Schalter S2 ausschalten.
13. Menüpunkt „3 = Prüfen CAS" anwählen.

```
auf CAS speichern

Startadr. : 8800
Endadr.   : 88ff
Name      : Test Aufzeichnung
```

Bild 12. Gleich beginnt die Aufzeichnung

14. Den Recorder starten.

15. Wenn keine Reaktion erfolgt, Schalter S1 umschalten und den Recorder zurückspulen. Neu starten.

16. *Bild 13* zeigt, wie es aussehen soll, wenn die Daten erfolgreich geladen wurden.

Wenn keine Reaktion erfolgt, kann man nochmals mit dem Skop-EPROM messen. Diesmal aber ohne den Teststecker. Auf dem Bildschirm sieht man dann die Signalform. Wenn kein Signal vom Recorder erscheint, sollte man zunächst alle Leitungen sorgfältig kontrollieren. Manchmal sind auch die Aufnahmesignale und Wiedergabesignale eines Recorders auf die gleiche Leitung gelegt. Man muß dann CAS-EIN mit CAS-AUS verbinden.

```
Prueflesen CAS

Test Aufzeichnung

OK M=Men
```

Bild 13. Das Bild zeigt einen gelungenen Test an

Jetzt wird programmiert

Nun zur Programmierung, denn jetzt kann man auch größere Programme schreiben, ohne daß sie nach dem Abschalten verloren sind.

Wenn man nach dem bisherigen Verfahren unterschiedlich große Quadrate programmieren wollte, dann müßte man für jedes Quadrat ein eigenes Programmstück schreiben. Das soll sich ändern. Dazu benötigen wir aber neue Befehle.

Lade indirekt

Im Befehl 2A xxxx.W steht xxxx für eine Zahl oder einen Namen, wie beim Lade-Wert-Befehl. Jetzt gibt die nachfolgende Zahl aber nicht den Wert an, der geladen werden soll, sondern sie ist eine Speicheradresse. Bei Ausführung des Befehls holt sich die CPU (Z80) von zwei aufeinanderfolgenden Speicherzellen, beginnend bei der angegebenen Adresse die Zahl, die dann geladen wird. Ein Beispiel zeigt schon alles:

```
8800:   2A 8900.W
        CD SCHREITE

8900:   #123.W
```

```
QUADRAT:=$
22 8900.W
21 #4.W
CD SCHLEIFE
2A 8900.W
CD SCHREITE
21 #90.W
CD DREHE
CD ENDSCHLEIFE
C9

VIELE:=$
21 #50.W
CD QUADRAT
21 #80.W
CD QUADRAT
21 #110.W
CD QUADRAT
21 #140.W
CD QUADRAT
C9
```

Bild 14. Viele Quadrate. Beachten Sie, daß in unserer Grafik-Sprache immer zuerst die Unterprogramme kommen und dann erst das Hauptprogramm, das diese Unterprogramme benutzt. Der Start des Gesamtprogrammes muß immer beim Hauptprogramm und nicht bei der niedrigsten Programmspeicherzellennummer liegen

Auf Adresse 8800 steht der Befehl „2A 8900.W". Wenn der Befehl ausgeführt wird, so wird der Inhalt der Speicherzellen 8900, 8901 geladen. Dort steht der dezimale Wert 123, wie die Anzeige ausweist. Also ist die Anweisung jetzt identisch mit dem Befehl „21 #123.W". Weshalb zwei Speicherzellen? Weil der Lade-indirekt-Befehl eine 16-Bit-Größe lädt, genauso, wie auch der Lade-Wert-Befehl. Wenn man einen Zahlenwert mit „.W" abschließt, werden automatisch zwei Speicherzellen verwendet. Deshalb kann man auch einfach #123.W als Datenwert angeben. „.W" steht für Wort. Entsprechend gibt es auch „.B" für Byte, wenn man nur genau ein Byte, also bei uns genau eine Speicherzelle belegen will. In einem Byte lassen sich 256 verschiedene Zahlen unterbringen, in einem Wort 65 536.

Jetzt benötigt man noch einen weiteren Befehl, um das ganze sinnvoll zu machen.

Der Speicher-Wert-Befehl

Im Befehl 22 xxxx.W ist xxxx wieder eine Adresse. Damit wird ein Wert, der vorher geladen oder errechnet wurde, auf die zwei Speicherplätze abgelegt, die durch xxxx bestimmt sind.

Ein vollständiges Programmbeispiel zeigt *Bild 14*. Achtung!: Die Lade-Indirekt-Adresse ist dort sedezimal angegeben, also ohne das Zeichen „#". *Bild 15* zeigt das Ergebnis des Programmlaufes auf dem Bildschirm. *Bild 16* zeigt den Speicherauszug als Kontrolle.

Hier noch ein Hinweis zu den Symbolen, also den Namen, den man bestimmten Adressen oder Zahlen gegeben hat.



Bild 15. Das Ergebnis von VIELE

Wenn man ein einzelnes Symbol löschen will, so kann man das mit folgender Anweisung tun:
SYMBOL:=%
(Anstelle von SYMBOL muß der zu löschende Name geschrieben werden.) Das Prozentzeichen ist dabei der Löschbefehl. Man verwendet ihn, wie das Dollarzeichen. Will man alle Symbole löschen, so kann man den Rechner kurz ausschalten.
Wie funktioniert das Programm aus Bild 14? Zunächst wird im Quadratprogramm mit dem Befehl „22 8900.W" ein Zahlenwert auf Adresse 8900 abgespeichert. Der Zahlenwert kann zum Beispiel durch einen normalen Ladebefehl, der vor Aufruf des Quadratprogramms ausgeführt sein muß, bestimmt werden.

Speicher ansehen

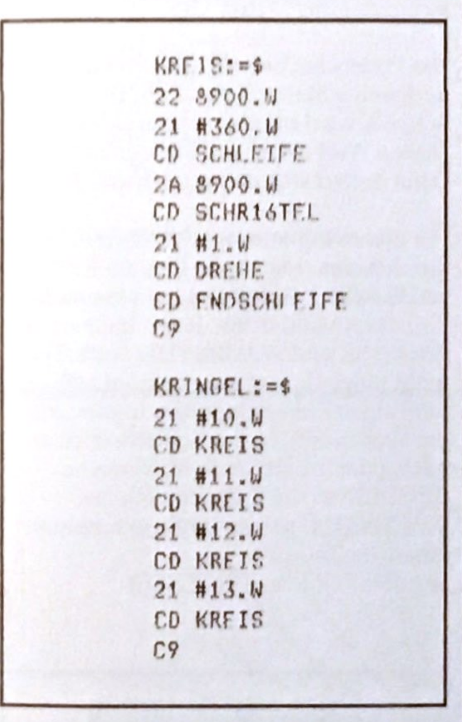
+=weiter -=rueckw R=Adr M=Menue

-- 0 1 2 3 4 5 6 7 8 9 A B C D E F

Bild 16. Der Speicherauszug zu Bild 14

Dann wird mit 21#4.W der Schleifenzähler geladen und der Schleifen-Befehl ausgeführt. Danach wird mit dem Lade-indirekt-Befehl „2A 8900.W" der Inhalt der Speicherzelle 8900 (und 8901) wieder geladen. Nun wird der Schreite-Befehl ausgeführt. Der Rest wie gehabt.

Im Programm „VIELE", das das eigentliche Hauptprogramm ist, wird zunächst mit dem Lade-Wert-Befehl die Seitenlänge eines Quadrates geladen und dann das Programm „Quadrat" aufgerufen. Der erste Befehl findet also eine sinnvolle Zahl vor, die Seitenlänge. Er sorgt dafür, daß dieser Parameter, so nennt man eine solche von Fall zu Fall die Größe des Quadrates bestimmende Zahl, richtig ins Unterprogramm kommt. So werden vier verschiedene Quadrate ausgeben. Man nennt das Zusammenspiel der Befehle zwischen Haupt- und Unterprogramm Parameterübergabe, da dem Unterprogramm „Quadrat" ein Parameter, die Seitenlänge des Quadrats, mitgegeben wird. Eigentlich sind Parameter ja nichts Neues mehr, denn der Schreite-, Drehe- oder Schleife-Befehl hat ja auch schon einen Parameter verwendet, nämlich die Schrittlänge, den Winkel oder die Anzahl der Schleifendurchläufe. Wichtig ist für Sie vielleicht bei unserem Sprachsystem, daß Sie fast immer die Unterprogramme vor dem Hauptprogramm im Speicher angelegt haben. Ein Programmlauf beginnt immer beim Hauptprogramm, also fast am Ende des ganzen Programmes! Jetzt kann man auch mal das Kreisprogramm als Unterprogramm verwenden. *Bild 17* zeigt ein Programmbeispiel, *Bild 18* das Ergebnis. Will man eine feinere Stufung haben, so kann man zum Beispiel mit 180 Schleifendurchläufen und 2° pro Drehung arbeiten – oder noch extremer.



```
KREIS:=$
22 8900.W
21 #360.W
CD SCHLEIFE
2A 8900.W
CD SCHR16TEL
21 #1.W
CD DREHE
CD ENDSCHLEIFE
C9

KRINGEL:=$
21 #10.W
CD KREIS
21 #11.W
CD KREIS
21 #12.W
CD KREIS
21 #13.W
CD KREIS
C9
```

Bild 17. Mehrere Kreise werden gezeichnet

Ins Innere des Z80

Der Z80 besitzt in seinem Inneren eine Reihe von speziellen Speicherzellen. Diese Speicherzellen nennt man Register. Der Z80 kann sich dort Werte merken. Der Befehl „21" lädt zum Beispiel einen Datenwert von außen in ein bestimmtes 16-Bit-Register im Inneren des Z80. Das Register besitzt einen Namen und heißt HL-Register. Warum gerade HL? Das hat der Hersteller einfach so definiert. Register besitzen also meist

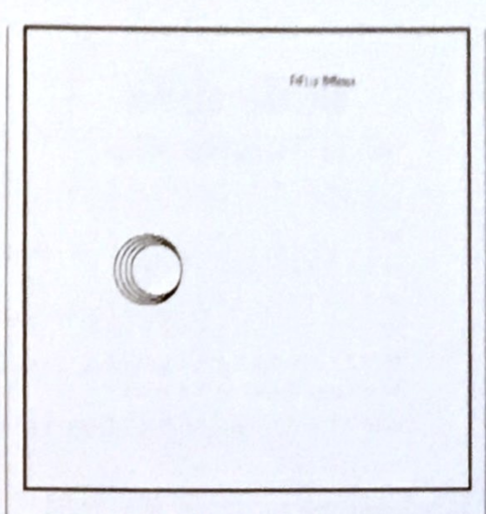


Bild 18. Hier sind es vier Kreise

Namen und keine Adressen, um sie leicht von normalen Speicherzellen unterscheiden zu können.

Der Z80 besitzt insgesamt sehr viele Register. Es sind dies neben HL noch BC, DE, HL', DE', BC', AF und AF' sowie IX, IY und SP, außerdem R und I und PC. Das Register PC ist der Programmzähler. Es bestimmt, von welcher Adresse im Speicher der nächste Befehl geholt wird. Am Anfang benötigt man noch nicht gleich alle Register, man kann sie nach und nach kennenlernen.

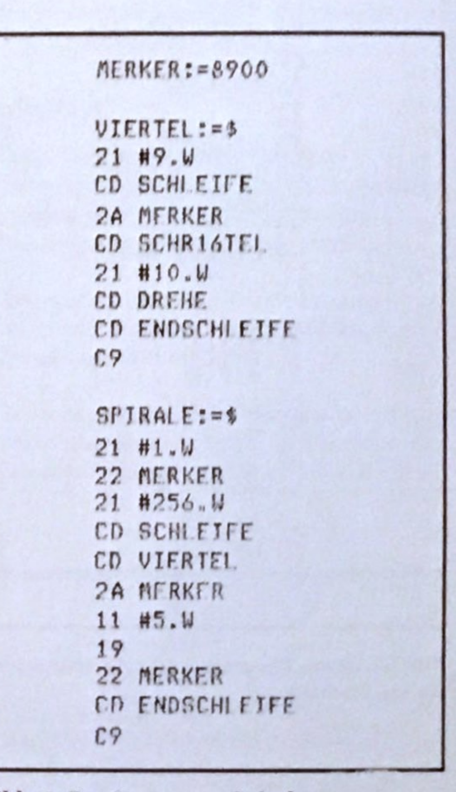
Der Z80 kann addieren

Zwei neue Befehle:
1. 11 xxxx.W
Der Befehlscode „11" ist der Lade-Wert-Befehl für das Register DE. Der Lade-Wert-Befehl „11" arbeitet wie der Befehl „21", nur daß der Zahlenwert nicht in HL, sondern in DE abgespeichert wird.
2. 19
„Addiere DE nach HL" heißt dieser Befehl, der keine weiteren Angaben benötigt. Das Ergebnis ist anschließend im Register HL zu finden.
Beispiel:
21 #5.W
11 #4.W
19
Nach Ausführung dieses Programms steht der Wert 9 im Register HL. Ein nachfolgender Befehl „CD SCHREITE" würde also 9 Schritte ausführen.

Dieser Addiere-Befehl ist interessant, wenn er in einer Schleife ausgeführt wird. Dazu ein Beispiel. Es soll eine Spirale gezeichnet werden.

Bild 19 zeigt das Programm. Bei jedem Schleifendurchlauf in SPIRALE wird der Inhalt der Speicherzelle „MERKER" (nach CD VIERTEL) um 5 erhöht. Damit wird der Viertelkreis immer größer und so ergibt sich eine Spiralform. Bei Beginn des Programms „SPIRALE" wird zunächst der Wert 1 geladen und in „MERKER" abgespeichert. Damit ist ein Startwert vorgegeben. Man nennt solch eine Vorbesetzung auch Initialisierung.

Beim Eingeben darauf achten, daß die Zuweisung „MERKER:=8900" nicht vergessen wird. Man könnte anstelle des Namens „MERKER" auch direkt die Zahl 8900 setzen. *Bild 20* zeigt die gezeichnete Spirale.



```
MERKER:=8900

VIERTEL:=$
21 #9.W
CD SCHLEIFE
2A MERKER
CD SCHR16TEL
21 #10.W
CD DREHE
CD ENDSCHLEIFE
C9

SPIRALE:=$
21 #1.W
22 MERKER
21 #256.W
CD SCHLEIFE
CD VIERTEL
2A MERKER
11 #5.W
19
22 MERKER
CD ENDSCHLEIFE
C9
```

Bild 19. Das Programm Spirale



Bild 20. Das Ergebnis von Spirale

```
GROESSE:=8900
WACHSTUM:=8902
WINKEL:=8904
ANZAHL:=8906

UNIVERS:=$
21 #36.W
CD SCHLEIFE
2A GROESSE
22 WACHSTUM
2A ANZAHL
CD SCHLEIFE
2A GROESSE
CD SCHREITE
2A WINKEL
CD DREHE
2A WACHSTUM
EB
2A GROESSE
19
22 GROESSE
CD ENDSCHLEIFE
2A WACHSTUM
22 GROESSE
CD ENDSCHLEIFE
C9

SPIRO:=$
21 #10.W
22 GROESSE
21 #120.W
22 WINKEL
21 #7.W
22 ANZAHL
CD UNIVERS
C9
```

Bild 21. Dieses Programm arbeitet sehr intensiv mit Parametern

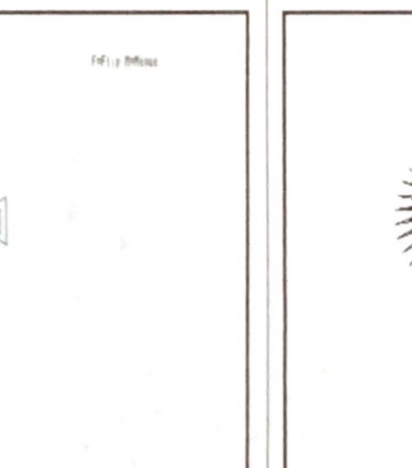


Bild 22. Ein kleines Kunstwerk, gefertigt vom Programm aus Bild 21

```
Speicher ansehen

+=weiter -=rueckw R=Adr M=Menue
-- 0 1 2 3 4 5 6 7 8 9 A B C D E F
[illegible]
```

Bild 23. Das ist der Speicherauszug

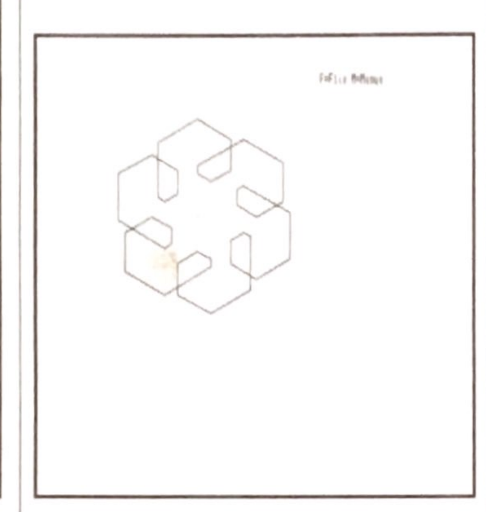


Bild 24. Nur ein Parameter in Spiro wurde geändert

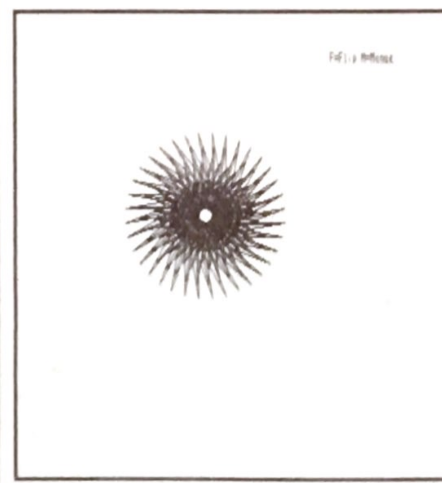


Bild 25. Noch eine Parameteränderung

Ein raffiniertes Programm

Für das nächste Beispiel wird ein neuer Befehl benötigt:
EB.
Damit wird der Inhalt der Registerpaare DE und HL vertauscht. *Bild 21* zeigt das Programm. Darin wird sehr intensiv mit Parametern gearbeitet. Im Programmteil „SPIRO“ werden die Startwerte festgelegt. Dann wird das Unterprogramm „UNIVERS“ aufgerufen. *Bild 22* zeigt, was passiert, wenn man es startet, *Bild 23* den Speicherauszug.

Wenn man die Parameter in SPIRO abändert, kann man andere Bilder erzeugen. Wenn man zum Beispiel den Winkel mit 60 vorbelegt, so entsteht *Bild 24*. Wenn man den Winkel mit 190 und die GROESSE mit 20 vorbelegt, so entsteht *Bild 25*. Wie funktioniert das Programm? *Bild 26* zeigt ein Struktogramm des Unterprogramms „UNIVERS“.

Das Programm besteht aus zwei ineinandergeschachtelten Schleifen. Die äußere Schleife wird hier 36mal durchlaufen. Diesen Wert kann man auch abändern. Dann ändert sich die gezeichnete Figur.

Die innere Schleife wird ANZAHL mal durchlaufen. Man hätte hier auch einen LADE-WERT-BEFEHL „21“ verwenden und die Anzahl direkt laden können. Bei GROESSE und WACHSTUM wäre das nicht möglich, denn sie werden während eines Programmlaufes immer wieder abgeändert. In der Schleife wird zunächst der Inhalt der Speicherzelle GROESSE in die Speicherzelle bei WACHSTUM gelegt. Im Struktogramm ist es die Zeile:
WACHSTUM := (GROESSE)

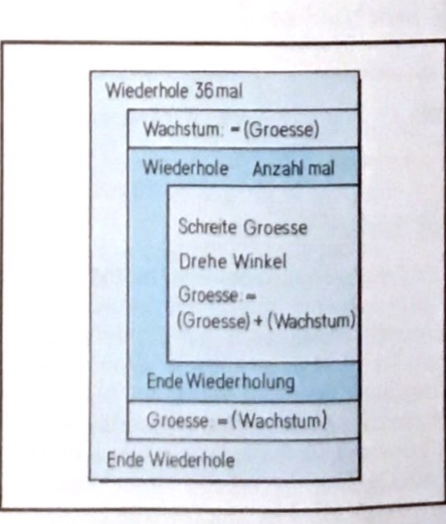


Bild 26. Das Struktogramm des Programms Univers aus Bild 21

21 xxxx.W	Lade-Wert-Befehl nach HL
2A xxxx.W	Lade-Indirekt-Befehl nach HL
22 xxxx.W	Speichere-Wert-Befehl von HL
11 xxxx.W	Lade-Wert-Befehl nach DE
19	Addiere DE nach HL
EB	vertausche HL mit DE
CD xxxx.W	Rufe Unterprogramm xxxx.W auf

Anstelle von xxxx.W kann auch ein Name stehen.

Bild 27. Das sind die bisher bekannten Befehle

```
21 #50.W
11 #10.W
EB
CD SCHREITE
C9
```

Bild 28. Programm zu Aufgabe 2

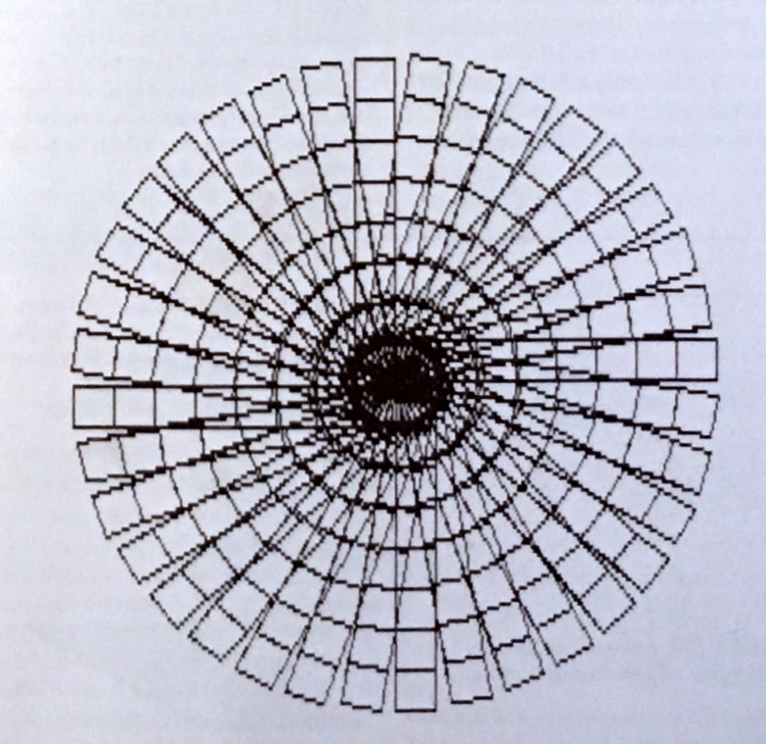
Die Klammern bedeuten, daß der Lade-Indirekt-Befehl verwendet wird. Denn es wird der Inhalt der Speicherzelle GROESSE und nicht die Adresse nach WACHSTUM gegeben. Dann folgt die innere Schleife. Dort wird um GROESSE weitergeschritten und um WINKEL gedreht. Dann folgt eine Addition. Es wird der Inhalt von GROESSE zum Inhalt von WINKEL addiert und in GROESSE abgespeichert. Nach Beenden der Schleife wird der Inhalt von WACHSTUM wieder in GROESSE gespeichert. Damit besitzt GROESSE den gleichen Inhalt, wie vor dem Start der Schleife.

Hier noch eine Bemerkung zur Zählschleife, wie sie auch im vorherigen Programmbeispiel vorkam. Mit einer Zuweisung in einem Struktogramm, wie MERKER := (MERKER) +5, wird der Inhalt der Speicherzellen bei der Adresse MERKER, um 5 erhöht. Führt man diesen Befehl in einer Schleife aus, so wächst die Zahl bei jedem Durchlauf um 5. Man sagt in diesem Fall auch, daß es sich um einen Zähler handelt.

Bild 27 zeigt die Befehle, die bis jetzt besprochen worden sind.

Aufgaben

1. Was tut der Befehl 11 #10.W?
2. Wieviel Schritte werden nach Ausführung des Schreitebefehls beim Programm aus *Bild 28* durchlaufen?
3. Entwerfen Sie ein Programm, das Quadrate mit der Seitenlänge 1 bis zur Seitenlänge 400 ausgibt?
4. Man entwerfe ein Programm, das Blumen in unterschiedlicher Größe auf dem Bildschirm ausgibt (BLUMENGARTEN).



```
8800:
VIELLEITER:=$
21 #36.W
CD SCHLEIFE
21 #8.W
CD SCHLEIFE
21 #4.W
CD SCHLEIFE
21 #20.W
CD SCHREITE
21 #90.W
CD DREHE
CD ENDSCHLEIFE
21 #20.W
CD SCHREITE
CD ENDSCHLEIFE
21 -#160.W
CD SCHREITE
21 #10.W
CD DREHE
CD ENDSCHLEIFE
C9
```

Rolf-Dieter Klein

Gut und Schlecht

Mikroelektronik, Folge 15

Jetzt können Sie schon Programme auf einem Massenspeichermedium ablegen und Unterprogramme programmieren. Abgesehen von viel Routine haben Sie schon das Wichtigste aus der Computerei kennengelernt. Trotzdem kennen Sie die meisten Dinge noch nicht. An einem Beispiel sollen jetzt einige weitere Befehle erklärt werden.

Elektronische Prüfanlagen spielen in der Industrie eine große Rolle. Dabei geht es zum Beispiel auch darum, Füllgut abzuwiegen oder Material nach Größe, Farbe und Gewicht oder anderen Kriterien zu sortieren.
Die Aufgabe soll es jetzt sein, eine solche Prüfanlage aufzubauen und zu programmieren. Dabei sollen Kästchen nach Gewicht sortiert werden. Der Einfachheit halber gebe es nur ein Kriterium, nämlich Kästchen schwer, oder Kästchen leicht. Das Gewicht wird mit einer Art Waage ermittelt. Kästchen, die zu schwer sind, sollen aussortiert werden. Dazu gebe es zwei Behälter. In den einen kommen die Teile, die gut sind, in den anderen die, die nicht gut sind.

Der Ansatz zur Lösung

Es sind zwei unterschiedliche Aufgaben zu lösen. Zum einen muß ein Gerät konstruiert werden, das die mechanische Aufgabe erfüllen kann. Dann muß ein Programm gefunden werden, das mit der Mechanik richtig zusammenspielt. Programmierung und mechanischer Aufbau greifen stark ineinander.

Die Mechanik

Die Messung des Gewichts kann man zum Beispiel mit einer Federwaage durchführen. Je schwerer ein Teil, desto mehr wird die Feder zusammengedrückt werden. Mit einem Tastschalter kann man feststellen, ob ein Teil zu schwer ist. Man benötigt aber auch einen Kontakt, der mitteilt, ob überhaupt ein Teil auf der Waage liegt. Damit kann man entscheiden, wann ein Sortiervorgang beginnen soll.
Einen schematischen Vorschlag zeigt *Bild 1*. Damit das Teil, das auf der Waage liegt auch nach links oder rechts ausgeworfen werden kann, gibt es noch einen Antrieb, der je nach Drehrichtung das Teil nach links oder rechts bewegt. Damit der Computer feststellen kann, wann der Antrieb die Nullstellung erreicht hat, wird wieder ein Tastschalter eingesetzt. Er gibt Kontakt, wenn der Auswurfarm zur Mitte der Waagschale zeigt.

Bild 1. Die Anordnung für die Experimente in diesem Kapitel

Im Bild 1 sind auch die beiden anderen Taster eingezeichnet. Der Liegt-auf-Kontakt wird immer dann betätigt, wenn ein Teil auf der Waage liegt, dabei muß er sowohl bei einem leichten Teil als auch bei einem schweren Teil Kontakt geben. Dazu dient eine längere Lamelle, die an der Waagstange befestigt wird. Eine kürzere Lamelle betätigt den Gewicht-zu-hoch-Kontakt, wenn die Feder zu stark zusammengedrückt wird.

Das Programm

Bild 2 zeigt ein Struktogramm. Dieses Struktogramm ist noch sehr grob. Der Programmteil zwischen „Wiederhole" und „unbegrenzt", wird unbegrenzt ausgeführt, denn der Prüfvorgang soll für immer laufen. In der ewigen Schleife wird abgefragt, „ist das Teil gut?". Wenn ja, so soll das Teil in die Gutschachtel, wenn nein, dann in den Abfall. Dieses Struktogramm ist noch sehr grob. Es kann daher im nächsten Schritt verfeinert werden.

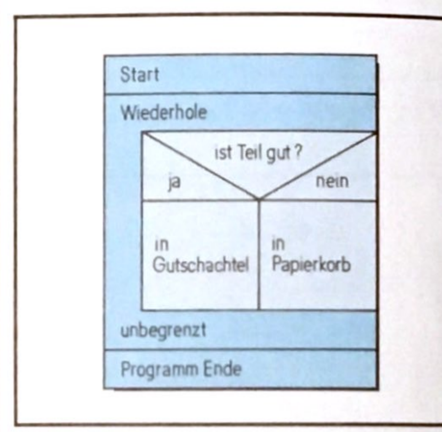


Bild 2. Ein Struktogramm kann einen Programablauf sinnfällig wiedergeben. Was hintereinander geschieht, steht im Struktogramm in rechteckigen Kästen. Was innerhalb einer Schleife zum Beispiel geschieht, ist „eingeschachtelt" im Kasten der Schleife. Fragen werden in ein Dreieck geschrieben, an dessen einer Seite der Ja-Teil angefügt ist, an der anderen der Nein-Teil. Was da getan werden soll, steht wieder in den Unterkästen. Der Programmende-Kasten wird hier nie erreicht werden, weil die Schleife nicht verlassen werden kann

Bild 3 zeigt das zweite Struktogramm. Im äußeren Block steht wieder „Wiederhole unbegrenzt". Dann folgt im nächsten Block eine Anweisung „Wiederhole bis Teil aufliegt". Denn vor dem Meßvorgang muß geprüft werden, ob ein Teil da ist.

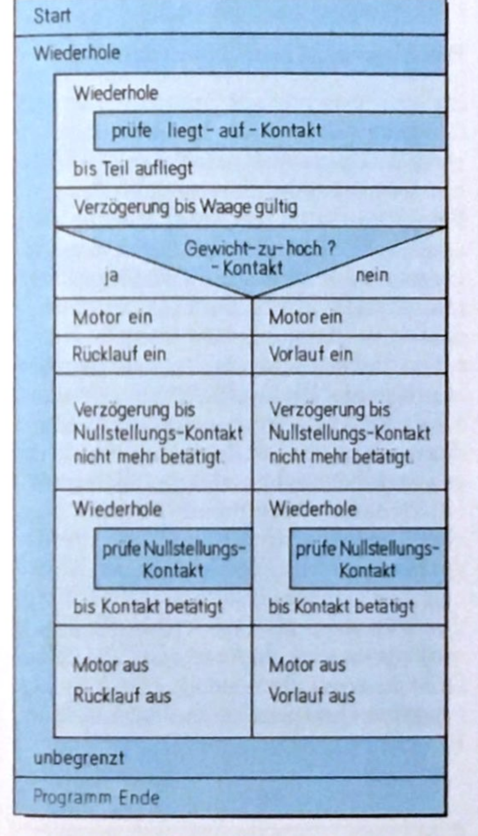


Bild 3. Das Struktogramm von Bild 2, wesentlich verfeinert. Die einzelnen Kästen hier sind Unterkästen der Kästen aus Bild 2

Dann muß eine kleine Zeit gewartet werden, bis der Meßvorgang abgeschlossen ist, denn wenn man ein Teil auf die Waage legt, so kann es passieren, daß zunächst versehentlich der Gewicht-zu-hoch-Kontakt betätigt wird und eine Fehlmessung wäre die Folge. Umgekehrt kann es auch sein, daß der Gewicht-zu-hoch-Kontakt erst nach dem Liegt-auf-Kontakt ausgelöst wird und der Rechner dann die Auswertung schon durchgeführt hat, ehe die Messung gültig war.

Erst nach einer Verzögerung folgt die Prüfung. Der Gewicht-zu-hoch-Kontakt wird abgefragt. Ist er betätigt, also das Gewicht zu hoch, so wird der linke Zweig ausgeführt, sonst der rechte. Beim linken Zweig wird zunächst der Motor eingeschaltet und dann der Rücklauf eingestellt. Beim rechten Zweig wird der Motor nach dem Einschalten auf Vorlauf gestellt. Damit wird die Unterscheidung beim Auswurf getroffen.
Dann folgt eine Verzögerung, denn der Motor bewegt den Auswurfarm nur langsam. Und es muß solange gewartet werden, bis der Nullstellungskontakt nicht mehr betätigt wird. Dann kann solange gewartet werden, bis er erneut Kontakt macht und damit der Auswurfarm eine volle Drehung durchgeführt hat. Wenn man die erste Verzögerung nicht geben würde, dann würde zwar der Motor gestartet, bevor der Arm sich aber noch weit bewegen könnte, würde er auch schon wieder abgestellt, denn der Nullstellungs-Kontakt wäre noch betätigt, da der Motor eine gewisse Trägheit besitzt. Nach einer Drehung wird der Motor dann wieder ausgeschaltet.

Eine Steuerschaltung

Nun zur Schaltungsbeschreibung. *Bild 4* zeigt die Schaltung. Die Transistor-Schaltung entspricht der Schaltung, die schon beim Roboter verwendet wurde. Man kann daher auch die Roboterplatte verwenden. Die Zuordnung zu den Ports der IOE-Karte ist gleich geblieben. Die IOE-Karte wird auf Adresse 30 gestellt, also die Brücken 7 und 6 eingelötet. Als Eingang sollen die Bits 0, 1, 2 des I/O-Ports mit Adresse 30 (I/O 0) verwendet werden. Damit kann es an die Programmierung gehen, doch fehlen noch ein paar Z80-Befehle um die Aufgabe zu lösen.

Weiter in den Z80

Der Z80 besitzt eigene Speicherzellen, das wurde schon gesagt. *Bild 5* zeigt nun alle schematisch auf. Dabei gibt es die Register A, B, C, D, E, H, L, dann A', B', C', D', E', H', L'. Jedes dieser Register kann 8 Bits speichern.
Einige Register können zusammengeschaltet werden, es sind BC, DE, HL, BC', DE' und HL'. Die haben Sie schon kennengelernt. Dann gibt es noch andere Register, wie zum Beispiel den Programmzähler. Dieses Register ist 16 Bit breit und läßt sich nicht teilen. Es beinhaltet immer die Adresse, des nächsten auszuführenden Befehls.
Das Register SP enthält den sogenannten Stackpointer. Darin werden Adressen gespeichert, die hauptsächlich für die Unterprogrammtechnik verwendet werden.
Das Registerpaar HL wurde bisher von unserer Grafik-Sprache verwendet. Mit dem Befehl „21 xxxx.W" wurde der Wert xxxx in das Registerpaar HL geladen. Beispiel: „21 ABCD.W" lädt den Wert AB in das Register H und den Wert CD in das Register L. Zahlreiche weitere Befehle gibt es, wenn man einzelne Register laden will. Hier ein kleiner Ausschnitt aus den Befehlen. Sie werden für das Waagenprogramm benötigt.

Wichtig sind da Befehle, die mit der Außenwelt zu tun haben. Zwei Befehle wurden schon in der Folge „Der Nichtstubefehl", Kapitel 6, verwendet. Der IN-Befehl (DB) und der OUT-Befehl (D3).
Die Codierung lautet:
DB xx.B für den IN-Befehl
und
D3 xx.B für den OUT-Befehl
Der IN-Befehl liest einen 8-Bit-Wert, ein Byte, in das Register A. Dabei steht xx für eine Adresse, die den Kanal angibt.

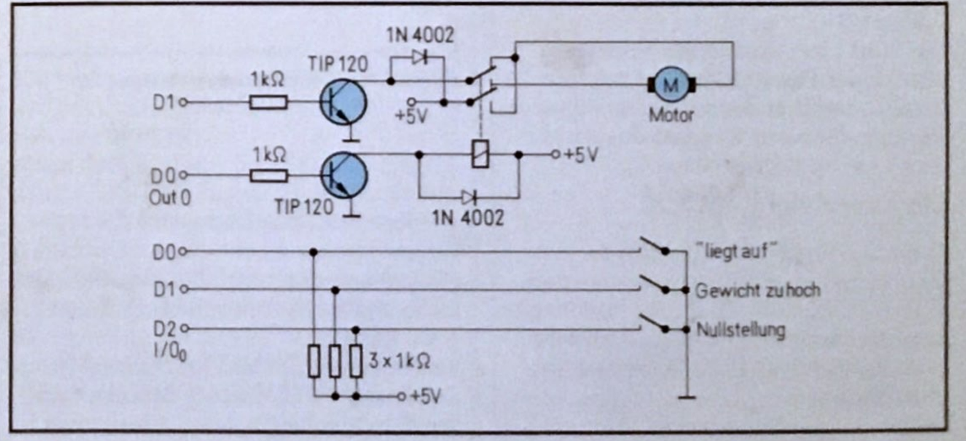


Bild 4. Alles, was die Auswerteelektronik benötigt, befindet sich auf der Roboter-Steuer-Platine. Die Anschlüsse links befinden sich auf der IOE-Platine

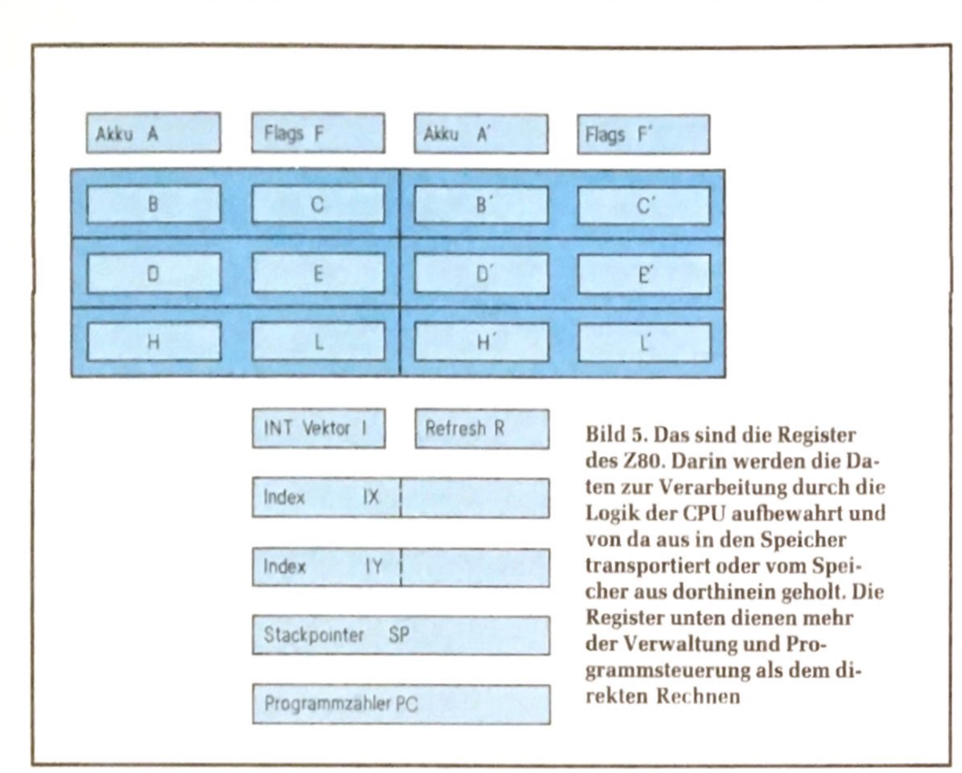


Bild 5. Das sind die Register des Z80. Darin werden die Daten zur Verarbeitung durch die Logik der CPU aufbewahrt und von da aus in den Speicher transportiert oder vom Speicher aus dorthinein geholt. Die Register unten dienen mehr der Verwaltung und Programmsteuerung als dem direkten Rechnen

Diese Adresse reicht von 0 bis FF, umfaßt also 256 Werte. Die IOE-Karte ist bei diesem Versuch auf die Adresse 30 (sedezimal 30) gelegt. Man kann durch den Befehl

DB 30.B

auf das Port I/O0 zugreifen. Da die IOE-Karte noch weitere Ports besitzt, sind noch andere Adressen gültig. Zum Beispiel wird mit dem Befehl DB 31 auf Port I/O1 zugegriffen. Das „.B" gibt an, daß der Wert „xx" ein Wert mit maximal 8 Bit, ein Byte, darstellt. Man nennt so etwas eine Typdeklaration.

Mit dem Befehl

D3 xx.B

wird ein Wert vom Register A zum Port mit der Adresse „xx" geschrieben.

Nun benötigen wir noch einen Ladebefehl für das Register A:

3E xx.B

Der Wert „xx" wird in das Register A übertragen. Dieser Befehl ist mit dem 21-Befehl vergleichbar, nur daß er ein anderes und kleineres Register anspricht.

Ein Experiment mit IOE

Dazu das Programm aus *Bild 6*.

Das Programm gibt wechselweise den Wert 00 und dann FF an das Port 30 aus.

1. e0 an Masse anschließen, damit die Ausgänge des 74LS374 freigegeben werden.
2. Programm eingeben und starten.
3. Mit dem Prüfstift an einem der Ausgänge von OUT0 messen.

Wenn man nicht schnell genug mit den Meßspitzen am Port ist, sieht man nichts, denn das Menü meldet sich gleich wieder. Beim Messen zeigt sich, daß die LEDs W1 und W2 kurz beide leuchten, wie auch die LEDs H und L. Hier könnte man einen neuen Befehl gut gebrauchen, der die Schleife unbegrenzt

```
START:=$
21 #10000.W
CD SCHLFIFE
3E 00
D3 30
3E FF
D3 30
CD ENDSCHLEIFE
C9
```

Bild 6. Das Testprogramm für Port 30

wiederholen hilft. Dann wird die Messung einfacher. Für solche und andere Fälle gibt es einen Befehl beim Z80, den unbedingten Sprungbefehl. Er lautet

C3 xxxx.W

und bedeutet: Springe zur Adresse xxxx. Der Zusatz .W deklariert, daß xxxx ein Wort ist, also 16 Bit. Mit C3 kann man eine Schleife konstruieren, die unbegrenzt läuft. Das modifizierte Programm zeigt *Bild 7*. Anstelle von xxxx.W kann auch ein Name stehen.

Wenn das Programm gestartet wird, kann die Messung nun in Ruhe ausgeführt werden. Will man das Programm anhalten, so muß man jetzt den RESET-Taster betätigen.

Neben dem Sprung C3 gibt es auch noch andere Sprungbefehle. Diese werden nur bei Erfüllung von bestimmten Bedingungen ausgeführt. Zwei davon sind für das Waagenprogramm wichtig. Der Befehl:

CA xxxx.W

Springe, wenn 0 nach xxxx, und der Befehl:

C2 xxxx.W

Springe, wenn nicht 0 nach xxxx.

Natürlich entsteht die Frage, was soll 0 oder nicht 0 sein?

Ein Einschub zum Nachdenken

Bisher wurde im Kurs ganz sanft mit der Computerei begonnen. Jetzt soll nicht verschwiegen werden, daß der Z80 einhundertachtundfünfzig verschiedene Befehle besitzt. Ein Spezialist muß sie nicht nur kennen, sondern auch kreativ verwerten können. Dies wird hier nicht gesagt, damit Sie erschrecken, sondern damit Sie erkennen, daß für Sie im Kurs schon viel getan worden ist. Nehmen Sie zum Beispiel die Grafik-Sprache. Wenn Sie da 21 #xxxx.W eingegeben haben, dann gelang das nur, weil das Grundprogramm für Sie sehr viel Arbeit geleistet hat. Es hat die Tastatur abgefragt, welches Zeichen getippt wurde, und dieses dann hergenommen und so in den Speicher geschrieben, daß daraus etwas für den Z80 verständliches wurde. Bei dieser Umsetzung haben fast alle Z80-Befehle mitgespielt, es hat also ein Z80-Programm kräftig mitgeholfen, ein Z80-Programm zu erzeugen.

```
START:=$
3E 00
D3 30
3E FF
D3 30
C3 START
```

Bild 7. Ein Testprogramm für Port 30 mit Endlosschleife

Besonders groß war die Mithilfe, wenn Namen im Spiel waren. Zum Beispiel mußte beim Tippen des Befehls CD SCHREITE im Grundprogramm zunächst einmal festgestellt werden, was SCHREITE bedeuten soll. Die zugehörige Zahl 3 wurde dann hergenommen und als 0300 in zwei aufeinanderfolgenden Speicherzellen nach dem CD notiert. Dies Ergebnis der Arbeit des Grundprogrammes, CD 0300, ist der eigentliche Z80-Befehl. Es ist purer Maschinencode, nichts weiter als eine Folge von Bitmustern von je acht Bit in 3 aufeinander folgenden Speicherzellen. Wenn der Mikroprozessor bei seiner Arbeit nun im Speicher auf den Befehl CD 0300 stößt, dann wird er durch das CD dazu gebracht, erst einmal die beiden nachfolgenden Bytes als Adresse herzuholen und dann als Unterprogrammstartadresse zu nehmen, um das fest im Grundprogramm vorprogrammierte Unterprogramm SCHREITE zu absolvieren.

Die Zeichensprache im Kurs hier hat also, damit unser Computer gleich etwas eindrucksvolles tun kann, ganze Unterprogramme als Befehle zu benutzen gestattet. So komfortabel ist der Z80-Befehlssatz natürlich nicht, in ihm gibt es keinen SCHREITE-Befehl. Vielmehr sind diese Befehle alle so unscheinbar, wie zum Beispiel einer der Ladebefehle, der „nur" etwas aus dem Speicher in ein Prozessor-Register transportiert. Der Grund, weshalb Sie im Kurs mit so komfortablen Unterprogrammen als Befehlen zu tun hatten, ist, daß es doch kaum Unterschiede in der Logik dabei gibt. Der SCHREITE-Befehl benötigte Angaben darüber, wie oft Schritte ausgeführt werden sollen und hat die automatisch auf dem Bildschirm ausgeführt. Jeder Maschinenbefehl des Z80 ist gleichsam ein in den Prozessor eingebautes superschnelles Unterprogramm, das etwas tut. Was, das ist von den Ingenieuren, die diesen Prozessor entworfen haben, exakt festgelegt worden und das kann man im Daten-Buch des Prozessors nachlesen. Einige der Befehle sind auch schon hier besprochen worden. Zum Beispiel der Lade-Indirekt-Befehl, der eine Angabe benötigt, von woher im Speicher er etwas holen soll und wohin es eingebaut hat, wohin er es bringen soll, nämlich in das interne Zielregister des Z80. Oder zum Beispiel der Additionsbefehl, der so gebaut war, daß er den Inhalt des Registerpaares DE hernahm und zum Inhalt des Registers HL addierte und das Ergebnis auch in HL niederlegte. Es ist mit diesen Beispielen auch gleich jeweils ein Beispiel für zwei wichtige Befehlstypen gegeben. Einmal der Lade-Befehl, der dem Datentransport dient und einmal der Additionsbefehl, der eine Operation mit den beteiligten Daten vornahm und aus zwei Registerinhalten die Summe machte.

Besonders oft wird beim Z80 das Register A benutzt, weil es viele Befehle gibt, die mit dessen Inhalt etwas tun. Zum Beispiel kann man .B-Daten bequem in A behandeln, dazu etwas addieren oder subtrahieren und viele andere interessante Dinge tun. Mit dem Register A steht das Register F in enger Verbindung. In ihm wird nämlich mitnotiert, ob zum Beispiel während einer Operation in A die Zahl für A zu groß geworden ist (Überlauf, Carry) oder ob nach einer Operation das Ergebnis Null geworden ist. Das ist wichtig, wie wir gleich am Beispiel unserer Sortiermaschine sehen werden.

Flags, Signale über den Zustand

Im F-Register wird unter anderem protokolliert, was an Besonderheiten bei Operationen im A-Register auftrat. Ein Befehl, eine Operation im A-Register, die beim Wägeproblem helfen soll, indem gezielt das F-Register damit in Abhängigkeit vom Inhalt von A geändert wird, sei jetzt besprochen.

Mit dem bedingten Sprungbefehl kann man dann entsprechend reagieren. Für die Waage wird also ein Befehl benötigt, der das Null-Flag verändert. Er lautet:

E6 xx.B,

was bedeutet: „verknüpfe den Inhalt des Registers A mit dem Wert xx durch UND". Ist das Ergebnis der UND-Verknüpfung = 0, dann wird das Null-Flag gesetzt, sonst wird es rückgesetzt.

Wozu die UND-Verknüpfung? Damit ist es möglich, einzelne Bits in einer 8-Bit-Gruppe herauszulösen und zu testen.

Beispiel:

Gegeben ist das Byte 5A im Register A. Es soll der Befehl E6 02 ausgeführt werden. Also steht nach der UND-Verknüpfung der Wert 02 im Register A, denn es gilt:

5A ist dual 01011010 und 02 ist dual 00000010. Beide werden Bit für Bit UND-verknüpft, es bleibt: 00000010, also der Wert 02.

Wenn im Register A der Wert 58 stehen würde, so würde sich ergeben: 01011000 UND 00000010 = 00000000 also der Wert 0.

So kann man durch die Verknüpfung mit den sogenannten Masken 01, 02, 04, 08, 10, 20, 40, 80, jedes einzelne Bit eines Bytes prüfen und erhält nach der UND-Verknüpfung einen Wert = 0 oder ungleich Null, je nach Zustand des Bits. Dieser Zustand wird in F notiert. Im Waageprogramm können wir die Kontakte mit diesem Befehl einzeln abfragen, die alle zusammen über die IOE-Platine ihre Stellung mitteilen.

```
START:=:
DB 30
E6 01
C2 START
3E FF
D3 30
3E 00
D3 30
C3 START
```

Bild 8. Ein Testprogramm mit bedingtem Sprung. Wenn das Ergebnis der Operation im Akkumulator Null hinterließ, dann Springe

Ein einfaches Programmbeispiel zeigt *Bild 8*.

```
Start:=$
Wdh1:=$
    Wdh2:=$
     DB 30.B
     E6 01.B
    C2 Wdh2
    21 5000.W
    CD Schleife
    00
    CD Endschleife
    DB 30.B
    E6 02.B
    CA Sonst
      3E 03.B
      D3 30.B
      21 5000.W
      CD Schleife
      00
      CD Endschleife
      Wdh3:=$
       DB 30.B
       E6 04.B
      C2 Wdh3
      3E 00.B
      D3 30.B
      C3 Endwdh
    Sonst:=$
      3E 01.B
      D3 30.B
      21 5000.W
      CD Schleife
      00
      CD Endschleife
      Wdh4:=$
       DB 30.B
       E6 04.B
      C2 Wdh4
      3E 00.B
      D3 30.B
    Endwdh:=$
C3 Wdh1
C9
```

Bild 9. Das ist schon ein etwas längeres Programm

1. Schaltung wie beim vorigen Beispiel.
2. Programm eingeben.
3. Programm starten.
4. Messen mit dem Prüfstift an einem der Ausgänge von OUT0. Es leuchten entweder die LED W1 oder W2.
5. Wenn man Bit 0 des Eingangs I/O0 (74LS245 am Leiterplattenrand), mit 0V verbindet, so erscheinen Pulse am Ausgang des 74LS374 (OUT0), also die LEDs W1 und W2 leuchten. Die Pulse sind solange vorhanden, bis der Eingang wieder von 0V getrennt wird.

Nun ist alles für das Waageprogramm bereit.

Das Waageprogramm

Bevor man das Programm (*Bild 9*) eingibt und startet, ist es gut, die Schaltung zu testen. Dazu rufe man das Menü „IO lesen" auf. Als Adresse wird 30 angegeben, die Adresse des Eingabeports.
Wenn man nun nacheinander die Kontakte betätigt, so kann man die einzelnen Bits auf dem Bildschirm beobachten.
Der Liegt-auf-Kontakt liefert eine 0 an der rechten Stelle des Bitmusters, wie *Bild 10* zeigt. Der Gewicht-zu-hoch-Kontakt liefert eine 0 an der daneben liegenden Stelle, also an Bit 1. Und der Nullstellungskontakt liefert eine 0 an Bit 2, wenn er betätigt wird. Über der dualen Darstellung, dem Bitmuster, ist auch der sedezimale Code des Zustandes ausgegeben, der hier aber nicht weiter interessiert.
Ehe man allerdings den Versuch durchführen kann, muß man noch die Taste „D" drücken, nachdem man die Adresse 30 und die Taste „CR" eingegeben hat. Nur so ist eine Dauer-Messung möglich, das Port wird dauernd abgefragt. Sonst würde nur eine einmalige Messung erfolgen.
Mit dem Menü-Punkt „IO setzen" kann der Motor getestet werden. Wenn man die Adresse 30 eingibt und den Wert 1 als Datenwert, so muß er sich in die eine Richtung drehen und wenn man die Adresse 30 und den Wert 3 angibt, dann in die entgegengesetzte Richtung. Die Verzögerung wird im Programm durch eine Schleife realisiert, in der ein NOP-Befehl (Code 00) ausgeführt wird. Wenn die Anzahl der Schleifendurchläufe sehr hoch ist, so ergibt sich dabei eine spürbare Verzögerung.

```
IO lesen
Adr 30
        FE
      11111110

R=Adr D=Dauer S=Stop M=Menue
```

Bild 10. In das Grundprogramm ist eine Portabfrage eingebaut, die den Zustand bitweise und sedezimal angibt

Ein Sprung nach vorn

Nun noch etwas zur Programmeingabe. Wenn man soweit ist, daß man den Befehl „CA Sonst" eingeben will, dann ergibt sich dabei eine Fehlermeldung: Das Grundprogramm erhöht die Adresse nicht weiter, denn der Name Sonst ist hier vor dieser Stelle noch nicht definiert worden. Die Definition erfolgt erst später im Programm. Man nennt so etwas eine Vorwärtsreferenz. Wenn man das Programm eingeben will, muß man zunächst alle Namen mit Vorwärtsreferenz weglassen und dafür beispielsweise „0.W" eingeben. Dann muß man, nachdem das Programm vollständig eingegeben wurde, wieder an den Anfang zurück und es erneut durchgehen um die Vorwärtssprünge einzugeben. Also hier bei den Befehlen „CA Sonst" und „C3 Endwdh". Beim zweiten Mal sind die Namen definiert und man kann sie jetzt verwenden. Dieses Verfahren wird in der Praxis immer angewendet, wenn Vorwärtsreferenzen vorkommen. Dazu ein anschauliches Beispiel:

In der Straßenbahn wird Fritzchen von einer Frau gefragt: „Ich muß bei der Luisenstraße aussteigen, wann ist das?" Darauf antwortet Fritzchen ganz ernst: „Wenn ich aussteige, so war es eine Station vorher." Die Frau muß also den Weg zweimal fahren, denn erst beim zweiten Mal kennt sie die Station von Fritzchen und damit auch ihre eigene Station. Im Computersprachgebrauch wird das Verfahren mit Zwei-Paß-Übersetzung bezeichnet.

Aufgabe

Das Programm Waage läßt sich noch abkürzen, wo?

Hinweis

Wenn man einmal zuviele Symbole definiert hat, so kann man einzelne Namen, die nicht mehr benötigt werden, auch löschen. Zum Beispiel

START:=%

löscht den Namen START. Das Prozentzeichen ist der Löschbefehl. Will man für eine neue Programmeingabe alle Symbole löschen, so schaltet man am besten den Rechner kurz aus.

```
8800:
  BEWEGEDREIECK:=$
  21 #150.W           CD DREHE
  CD SCHLEIFE         CD WAIT
  21 -#90.W           3E 01.B
  CD DREHE            D3 70.B
  21 #101.W           21 #100.W
  CD SCHREITE         CD SCHREITE
  21 #120.W           CD WAIT
  CD DREHE            3E 00.B
  21 #100.W           D3 70.B
  CD SCHREITE         21 #210.W
  21 #120.W           CD DREHE
                      CD ENDSCHLEIFE
                      C9
```


Rolf-Dieter Klein

Schrecksekunde

Mikroelektronik, Folge 16

Es muß betont werden, daß Sie, wenn Sie bis hierher vorgedrungen sind, schon sehr viel gelernt haben. Im Grunde haben Sie schon alles gelernt, was man über das Funktionieren von Computern wissen muß. Trotzdem sind Sie erst ein ganz kleines Stück in das gewaltige Gebiet eingedrungen, denn es gibt so viele Möglichkeiten, daß man tatsächlich einige Zeit benötigt, um einen ersten angemessenen Überblick zu bekommen. Ein paar Monate Beschäftigung mit dem Thema sind zuwenig. Deshalb wird ihnen auch dieses Kapitel etwas Kopfschmerzen bringen, jedenfalls am Ende. Da bekommen Sie ein erstes richtiges Maschinenprogramm vorgesetzt. Versuchen Sie es zu verstehen, mit dem Begleitbuch in der Hand oder mit einem anderen Buch, das den Z80 zum Thema hat. Wir meinen, daß Sie jetzt die ersten eigenen Schritte wagen müssen – hinein ins kalte Wasser. Wenn wir Ihnen alles hier erzählen sollten, dann können Sie solch eine dicke Schrift gar nicht mehr tragen.

Einen Computer kann man auch für genaue Zeitmessungen einsetzen. Will man eine Zeit bestimmen, so gibt es verschiedene Möglichkeiten. Zum einen kann man eine Verzögerungsschleife als Zeitbasis einsetzen und dann ein Register hochzählen, daß dann die jeweilige Zeiteinheit mißt, oder man verwendet einen externen Takt, der sehr genau ist und zählt diesen.
Als genaue Zeitschleife läßt sich die Sequenz aus *Bild 1* verwenden. Dabei benötigt der Aufruf dieses Programms 10 Millisekunden. Ein Schleifendurchlauf benötigt 6 Mikrosekunden, wenn die CPU mit einem Takt von 4 MHz betrieben wird.
Die Gesamtdauer ist nicht ganz exakt angegeben, da Aufruf und andere Befehle auch Zeit benötigen. Doch damit kann man schon eine kleine Uhr bauen.
Bild 2 zeigt ein Programmbeispiel.
Das Unterprogramm „Milli20" dient als Zeitmesser. *Bild 3* zeigt die Gestaltung des Zeigers. Das Hauptprogramm „UHR" gibt zuerst den Zeiger aus, dann führt es die Milli20-Routine 50mal aus. Damit wird etwa 1 Sekunde verbraucht. Dann wird der Zeiger gelöscht und die Schildkröte um 6 Grad im Uhrzeigersinn gedreht, denn 60mal 6 Grad ergibt 360 Grad, also pro Minute eine Umdrehung für den Zeiger.

Im Unterprogramm Zeiger finden sich ein paar neue Befehle. Mit 32 xxxx.W kann man den Inhalt des Registers A abspeichern. Mit AF wird das Register A auf 0 gesetzt. Mit den Befehlen wird erreicht, daß die Schildkröte nicht mehr sichtbar ist. Mit der Ausgabe auf den Port 60 wird erreicht, daß auch wirklich die Seite 0 angezeigt wird, denn die Uhr soll nicht in einer unsichtbaren Seite stehen.
Im Hauptprogramm gibt es noch eine Ausgabe auf das Port 70, das ist der Grafikprozessor. Wird dort der Wert 1 ausgegeben, so wirken alle nachfolgenden Befehle löschend. Wird der Wert 0 ausgegeben, so wirken alle nachfolgenden Zeichenbefehle schreibend, wie man es gewohnt ist.

Uhr mit externem Takt

Ein externer Takt. Einen solchen Takt gibt es und der kommt vom Grafikprozessor selbst. Dieser liefert nämlich einen 20-ms-Takt, den man abfragen kann. Dieser Takt wird normalerweise zum Aufbau des Bildes auf dem Bildschirm gebraucht, denn es ist die Bildwiederholrate.
Bild 4 zeigt ein Programm, das dies ausnutzt. Die Uhr geht viel genauer, da ein absoluter Zeitmaßstab verwendet wurde. Nun kann man die Uhr noch um ein Gehäuse ergänzen. Dazu ein Beispiel in *Bild 5*. *Bild 6* zeigt das Ergebnis auf dem Bildschirm.
Die größe der Schrittweite für den Kreis errechnet sich aus:

2 · PI · RADIUS/360.

Es ergibt sich: 3.49.
Damit der Kreis genauer angepaßt werden kann, wird das Unterprogramm SCHR16TEL verwendet, dann muß man das Ergebnis mit 16 multiplizieren, somit ergibt sich:

55.85

oder gerundet:

56

also 56 Schritte.
Man kann das Programm jetzt noch erweitern, z. B. um einen Minuten- und Stundenzeiger, dazu muß man nur eine Schleife mit 60 um das alte Programm legen und den letzten Sprung weglassen.

```
MILLI20:=$
  01 #3333.w    xxxx ist der Zähler pro 6 Mikrosekunden
 WDH:=$         er wird in BC geladen
  0B            Verringere BC um eins
  78            Lade B nach A
  B1            verknüpfe ODER mit C
  C2 WDH        wenn nicht 0, dann zurück nach WDH
  C9            Ende
```

Bild 1. Ein Zähler in Software aufgebaut. Die Eingabe bitte wie üblich mitändern

```
ZEIGER:=$          Uhrzeiger fuer Sekunden als Unterprog.
 21 #3.W
 CD DREHE          Zeiger mit bel. Form
 21 #100.W
 CD SCHREITE
 21 -#6.W
 CD DREHE
 21 #100.W
 CD SCHREITE
 21 #6.W
 CD DREHE
 21 -#100.W
 CD SCHREITE
 21 -#6.W
 CD DREHE
 21 -#100.W
 CD SCHREITE
 21 #3.W
 CD DREHE
 AF                Flip auf 0 setzen
 32 8048           damit keine Bildstörung
 CD WAIT
 AF
 D3 60             Seitenregister auf 0 setzen
 C9
```

Bild 2. Ein Minutenzeiger bewegt sich

```
UHR:=$
 CD ZEIGER         erst mal Zeiger zeichnen
 21 #50.W          50 x 20 ms = 1 sek warten
 CD SCHLEIFE
 CD MILLI20
 CD ENDSCHLEIFE
 CD WAIT           Warten bis Graphikprozessore fertig
 3E 01             Loeschfunktion einschalten
 D3 70             Und aktivieren
 CD ZEIGER         alten Zeiger loeschen
 CD WAIT           Warten bis fertig
 3E 00             und wieder Schreibfunktion setzen
 D3 70
 21 -#6.W          Im Uhrzeigersinn um 1sek drehen
 CD DREHE
 C3 UHR            und dann wieder Zeiger ausgeben
```

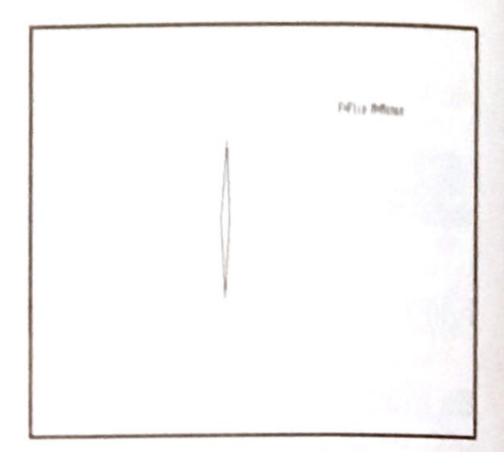
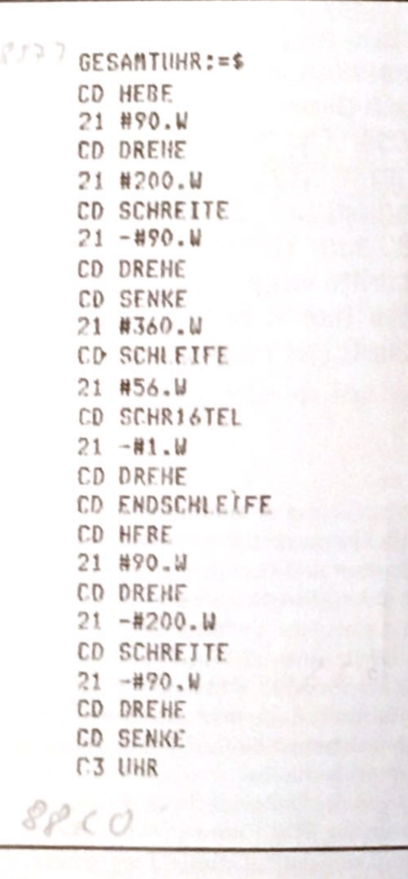


Bild 3. Das ist der Zeiger



```
GESAMTUHR:=$
 CD HEBE
 21 #90.W
 CD DREHE
 21 #200.W
 CD SCHREITE
 21 -#90.W
 CD DREHE
 CD SENKE
 21 #360.W
 CD SCHLEIFE
 21 #56.W
 CD SCHR16TEL
 21 -#1.W
 CD DREHE
 CD ENDSCHLEIFE
 CD HEBE
 21 #90.W
 CD DREHE
 21 -#200.W
 CD SCHREITE
 21 -#90.W
 CD DREHE
 CD SENKE
 C3 UHR
```

Bild 5. Die ganze Uhr mit Ziffern-Blatt

Ein Reaktionstest

Hier soll eine Warteschleife zur Zeitmessung verwendet werden. Auf dem Bildschirm erscheint zunächst die Meldung „GRUEN". Wenn dann plötzlich die Schrift „ROT" erscheint, so soll so schnell als möglich eine Taste gedrückt werden. Die Zeit zwischen Erscheinen des Textes „ROT" und dem Drücken der Taste soll gemessen werden. Wenn die Taste zu früh gedrückt wird, so erscheint die Meldung „PFUI" auf dem Bildschirm. Das Programm dazu ist sehr umfangreich und in *Bild 7* gezeigt.
Die Taste wird an den Eingang „INT" der CPU angeschlossen. Diese Leitung findet sich auf dem Bus. Die Taste wird zwischen dem Eingang und 0 V geschaltet. *Bild 8* zeigt die Schaltung. Dieser Eingang bewirkt etwas ganz besonderes. Wenn ein 0-Signal anliegt, so wird die Programmausführung unterbrochen und ein sogenanntes Interruptprogramm ausgeführt. Damit ist es möglich, z. B. die Zeitschleife sofort abzubrechen. Um den Eingang verwenden zu können müssen zwei Befehle gegeben werden:
Einmal FB (Enable Interrupt), damit wird der Interrupteingang freigeschaltet und zum anderen der Befehl ED 56 oder IM 1 mit dem wird bestimmt, daß bei einer Unterbrechung, auf Adresse 38h die Programmausführung fortgesetzt wird. Dort steht aber ein Sprung, denn die Stelle liegt im Grundprogramm. Der


```
UHR:=$
 CD ZEIGER         erst mal Zeiger zeichnen
 21 #50.W          50 x 20 ms = 1 sek warten
 CD SCHLEIFE
WA1:=$             Warten auf 20ms Puls
 DB 70             Liegt an Port 70
 E6 02             Bit 1 an
 C2 WA1            ungleich 0 dann zurueck
WA2:=$             warten bis wieder 0
 DB 70             sonst Störungen
 E6 02
 CA WA2
 CD ENDSCHLEIFE
 CD WAIT           Warten bis Graphikprozessore fertig
 3E 01             Loeschfunktion einschalten
 D3 70             Und aktivieren
 CD ZEIGER         alten Zeiger loeschen
 CD WAIT           Warten bis fertig
 3E 00             und wieder Schreibfunktion setzen
 D3 70
 21 -#6.W          Im Uhrzeigersinn um 1sek drehen
 CD DREHE
 C3 UHR            und dann wieder Zeiger ausgeben
```

Bild 4. Eine Uhr, die vom Fremdimpulsen an Port 70 gesteuert wird

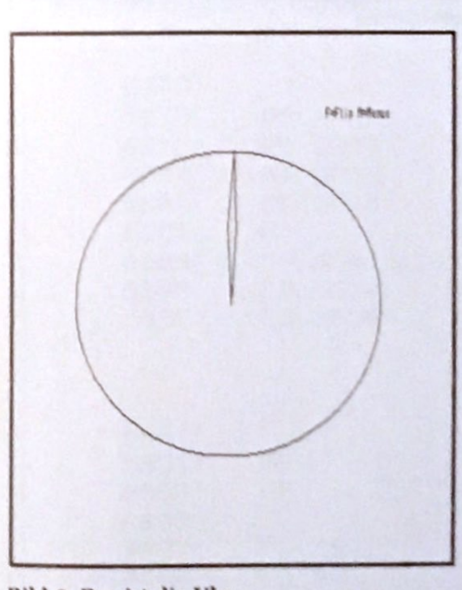


Bild 6. Das ist die Uhr

Sprung führt auf Adresse 8003 und das ist im RAM. Wenn man dort einen Sprung auf die eigene Routine setzt, so wird beim Interrupt die Ausführung dort hin geleitet.

Nun zur Eingabe des Programms. Das *Bild 7* zeigt einen Programmausdruck in noch unbekannter Form. Es ist ein Assemblerausdruck. Dabei wurde die Übersetzung der Befehle in den Maschinencode durch ein Assembler-Programm vorgenommen. Wenn man das Beispiel eintippen will, so gibt man einfach die Codezeilen ein, z. B. bei 8829 steht:

8829'   32 8003   ld (8003),a

Man tippt:

32 8003.W

Immer wenn eine Zahl auftaucht, die 16 Bit groß ist, gibt man beim Grundprogramm die Zahl mit einem nachfolgenden „.W" ein. Einzelne 8-Bit-Größen kann man direkt eingeben.
Wenn in der rechten Spalte z. B. „start1:" steht, so kann man auch „start1:=$" eingeben um einen Namen zu definieren, damit wird die Fehlersuche erleichtert. Das Programm wird auf Adresse 8827 oder beim Namen „start1" gestartet.
Im Programm werden einige neue Z80-Befehle verwendet, die man im Buch „Mikrocomputer selbstgebaut und programmiert findet". Ferner aber auch neue Unterprogramme aus dem Grundprogramm, z. B. CD WRITE oder „call write", wie es im Text steht. Damit kann man Texte auf dem Bildschirm ausgeben, z. B. den Text beim Namen „MSG1".
Dort steht zunächst dw 20,100. Eingegeben wird der Codeteil: „0014.W" und „0064.W". Der erste Wert ist die x-Koordinate, der zweite die y-Koordinate. Dann folgt der Wert „44h", das ist die Schriftgröße. Der nachfolgende Wert bestimmt noch, daß die Schrift senkrecht stehend ausgegeben wird. Das sind alles Angaben für die GDP64. Dann folgt der Text, diesen kann man auch eingeben, wenn man „GRUEN" in doppelten Anführungszeichen tippt.
Die doppelten Anführungszeichen am Anfang und Ende darf man bei der Eingabe nicht vergessen. Dann folgt noch der Wert 0, damit wird dem Grundprogramm gesagt, daß der Text zu Ende ist. Beim Aufruf lädt man in das Registerpaar HL die Anfangsadresse des Textblockes, also hier die Adresse bei dem Namen „MSG1".
Es kostet Sie bestimmt viel Mühe, dieses Programm zu durchschauen. Nehmen Sie ein Z80-Datenbuch und spüren Sie die Wirkung aller Befehle auf. Wir können Ihnen nur am Ende des Heftes eine Kurzfassung bieten, denn sonst würde das ganze mehr als ein Buch. Wenn Sie bis hierher durchgehalten haben, dann lohnt es sich ohnehin für Sie, sich Bücher anzuschaffen.
Es wäre ganz schön, das Programm auch dauerhaft zu speichern. Dazu kann man den Kassettenrecorder verwenden, oder einen EPROM-Programmierer. Der EPROM-Programmierer ist ebenfalls als Bausatz verfügbar. Beim Aufbau hält man sich am besten an die beigefügte Bauanleitung oder geht nach dem Buch „Mikrocomputer selbstgebaut..." vor. Wichtig ist, daß der EPROM-Programmierer auf 50 ms abgeglichen wird. Dazu kann man ein Oszilloskop verwenden, oder ein anderes Hilfsmittel. Oder man schreibt sich eine Routine, die die Zeit mißt und anzeigt, denn daß muß mit dem jetzigen Wissensstand möglich sein. Mit dem PROMMER kann man den Bereich 8800 bis 8FFF auf EPROM brennen (z. B. ein 2716) und anstelle des RAM-Bausteins in Fassung 3 einsetzen. Das geht, da daß Reaktionsmeßprogramm den RAM-Bereich bei 8700 als Merker nützt und der Bereich bleibt ja erhalten.

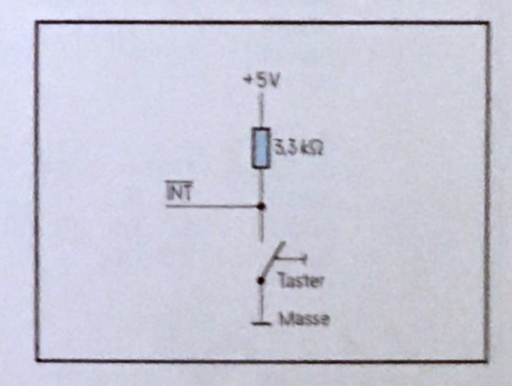


Bild 8. Der Taster wird angeschlossen

```
                        z80

                        ; Interrupt-Uhrenprogramm

001E                    write equ 1eh
003B                    wait equ 3bh
0030                    clr equ 30h
0070                    gdp equ 70h
0024                    ci equ 24h
0018                    moveto equ 18h

8700                    merker equ 8700h
87FF                    stack equ 87ffh


                        org 8800h
8800'                   start:
8800'    C3 8827'       jp start1

8803'                   msg1:
8803'    0014 0064      dw 20,100
8807'    44 00          db 44h,0
8809'    47 72 75 65    db 'Gruen',0
880D'    6E 00

880F'                   msg2:
880F'    0014 0064      dw 20,100
8813'    44 00          db 44h,0
8815'    52 6F 74 20    db 'Rot  ',0
8819'    20 00

881B'                   msg3:
881B'    0014 0064      dw 20,100
881F'    44 00          db 44h,0
8821'    50 66 75 69    db 'Pfui ',0
8825'    20 00



8827'                   start1:
8827'    3E C3           ld a,0c3h
8829'    32 8003         ld (8003h),a
882C'    21 886A'        ld hl,intprog
882F'    22 8004         ld (8004h),hl
8832'    31 87FF         ld sp,stack
8835'    ED 56           im 1
8837'    3E 00           ld a,0
8839'    32 8700         ld (merker),a   ; pfui-Flag
883C'    FB              ei
883D'    CD 0030         call clr
8840'    CD 0024         call ci ; warten auf Startanforderung
8843'    21 8803'        ld hl,msg1
8846'    CD 001E         call write
8849'    ED 57           ld a,i
```

Bild 7. Dieses Programm wurde mit einem Grundprogramm geschrieben, das von den Profis benutzt wird. Es heißt Assembler. Beginnen Sie zunächst mit der Analyse bei 8827. Davor liegen Speicherplätze, die Daten und Texte enthalten. Allerdings liegt auf Adresse 8800 ein JMP im Programm. Ganz am Anfang oben sind Vereinbarungen getroffen. „write" soll zum Beispiel 1Eh, also die Adresse 1E sedezimal, sein. Hinten ist nochmals die Tabelle der Symbole angefügt, weil auch im Programmtext noch Vereinbarungen vorkommen


```
884B'    E6 3F           and 3fh
884D'    C6 0A           add a,10
884F'    67              ld h,a
8850'                   lp2:
8850'    CD 88BB'        call del100
8853'    25              dec h
8854'    20 FA           jr nz,lp2
8856'    21 880F'        ld hl,msg2
8859'    CD 001E         call write
885C'    3E 01           ld a,1
885E'    32 8700         ld (merker),a
8861'    11 0000         ld de,0 ; Zeitzaehler
8864'                   miss:
8864'    CD 88C7'        call ms1
8867'    13              inc de
8868'    18 FA           jr miss


886A'                   intprog:
886A'    3A 8700         ld a,(merker)
886D'    FE 01           cp 1
886F'    28 0C           jr z,ok
8871'    21 881B'        ld hl,msg3
8874'    CD 001E         call write
8877'    CD 0024         call ci
887A'    C3 8800'        jp start


887D'                   ok:               ; de=timer umrechnen
887D'    D5              push de                        und ausgeben
887E'    21 0014         ld hl,20
8881'    11 0014         ld de,20
8884'    CD 0018         call moveto      ; position bestimmen
8887'    E1              pop hl           ; Ziel zeit in ms
8888'    CD 8891'        call ausgabe
888B'    CD 0024         call ci
888E'    C3 8800'        jp start



8891'                   ausgabe:
8891'    AF              xor a
8892'    F5              push af ; als flag
8893'                   aus1:
8893'    CD 88A7'        call div10
8896'    F6 30           or 30h
8898'    F5              push af
8899'    7C              ld a,h
889A'    B5              or l
889B'    20 F6           jr nz,aus1
889D'                   aus2:
889D'    F1              pop af
889E'    B7              or a
889F'    C8              ret z
88A0'    CD 003B         call wait
88A3'    D3 70           out (gdp),a ; Ausgabe
```

```
Macros:

Symbols:
8893'   AUS1        889D'   AUS2        8891'   AUSGABE
0024    CI          0030    CLR         88BB'   DEL100
88A7I'  DIV10       88B8'   DIV2        88AA'   DIVLP
0070    GNP         886A'   INTPROG     88BE'   LP
8850'   LP2         8700    MERKER      88C4'   MISS
0018    MOVETO      88C7'   MS1         88CA'   MS2
8803'   MSG1        880F'   MSG2        881B'   MSG3
887D'   OK          87FF    STACK       8800'   START
8827'   START1      003B    WAIT        001C    WRITE
```

```
88A5'   18 F6            jr aus2


88A7'                   div10::   ; hl /10 -> hl rest a
88A7'   AF               xor a
88A8'   06 10            ld b,16 ; 16 bits relevant
88AA'                   divlp:
88AA'   CB 25            sla l
88AC'   CB 14            rl h
88AE'   CB 17            rl a
88B0'   FE 0A            cp 10
88B2'   38 04            jr c,div2
88B4'   D6 0A            sub 10
88B6'   CB C5            set 0,l
88B8'                   div2:
88B8'   10 F0            djnz divlp
88BA'   C9               ret


88BB'                   del100:
88BB'   11 0064          ld de,100
88BE'                   lp:
88BE'   CD 88C7'         call ms1
88C1'   1B               dec de
88C2'   7B               ld a,e
88C3'   B2               or d
88C4'   20 F8            jr nz,lp
88C6'   C9               ret

88C7'                   ms1:
88C7'   01 00A6          ld bc,1000/6
88CA'                   ms2:
88CA'   0B               dec bc
88CB'   78               ld a,b
88CC'   B1               or c
88CD'   C2 88CA'         jp nz,ms2
88D0'   C9               ret

                        end
```


Rolf-Dieter Klein

Das ist GOSI

Eine höhere Programmiersprache

GOSI ist der Sprache LOGO entlehnt, und zwar der deutschen IWT-Version. Sie enthält viele Sprachelemente daraus, allerdings keine Listenverarbeitung. Mit GOSI lassen sich alle Merkmale einer höheren Programmiersprache zeigen, wie strukturierte Programmierung, Prozeduren und Parameterkonzepte. Der Übergang von unserer Zeichensprache zu einer höheren Programmiersprache läßt sich durch Gegenüberstellung von Programmbeispielen zeigen.

GOSI ist die Abkürzung für „Graphisch Orientierte Sprache I". Die Sprache GOSI ist verwandt mit der Sprache LOGO, die besonders leicht zu erlernen ist. GOSI läuft auf der SBC2-Karte anstelle des Grundprogramms. Nach dem Einschalten meldet sich GOSI, wie in *Bild 1* gezeigt.

```
GOSI - Graphisch orientierte Sprache I
(C) Muenchen 1984 Rolf-Dieter Klein Vers 1.1
"Ein Hauch von LOGO ..."
```

Bild 1. So meldet sich GOSI

GOSI enthält eine Schildkröte, die man bewegen kann, genau wie bei der Zeichensprache. Doch die Befehle sind jetzt einfacher. Man muß keinen Maschinencode mehr eingeben.

Erstes Beispiel:
Für die Vorwärtsbewegung gibt es den Befehl

VORWAERTS

Wieviel geschritten werden soll, wird durch eine Zahl definiert, die man hinter den Befehl tippt (durch ein Leerzeichen trennen).

VORWAERTS 100

Wenn man dann die Taste CR betätigt, so wird der Befehl ausgeführt und es erscheint eine Linie auf dem Bildschirm. Bei der Zeichensprache mußte man die Sequenz:

21 #100.W
CD SCHREITE

programmieren, um das gleiche Ergebnis zu erhalten. Die höhere Programmiersprache ist also in der Lage, durch einen Befehl die Ausführung mehrerer Maschinenbefehle zu erzwingen. Der Übersetzer, d. h. das Programm, das die Eingabe in eine Befehlssequenz umsetzt, kann je nach Konzeption unterschiedlich arbeiten. Beim sogenannten Interpreter wird ein Befehl eingelesen, wie z. B. VORWAERTS 100 und dann sofort ausgeführt; dann wird der nächste Befehl eingelesen, wieder analysiert und ausgeführt.

Bei einem sogenannten Compiler wird erst einmal alles eingelesen und dann zunächst komplett in eine Maschinensequenz umgesetzt. Erst, wenn das ganze Programm fertig als Maschinencode dasteht, wird es ausgeführt.

Die Programmiersprache GOSI ist ein Interpreter, denn ein Interpreter besitzt den Vorteil, daß man auch einmal einen einzelnen Befehl per Tastatur eingeben kann und sofort das Ergebnis auf dem Bildschirm sieht. Bei einem reinen Compiler müßte man zunächst das ganze Programm eintippen und könnte erst nach der vollständigen Übersetzung das Ergebnis betrachten. Interpreter arbeiten dafür aber im allgemeinen ein Programm langsamer ab als ein von Compilern erzeugtes Maschinenprogramm, da sie alle Befehle immer wieder von neuem übersetzen müssen.

Einige Befehle von GOSI

Neben dem VORWAERTS-Befehl gibt es auch: RUECKWAERTS

Alle Befehle kann man auch abkürzen:

VORWAERTS – VW
RUECKWAERTS – RW

Zum Drehen der Schildkröte gibt es:

LINKS – LI
RECHTS – RE

dann

STIFTAB – SA
STIFTHOCH – SH

und

SCHR16TEL – entsprechend der Zeichensprache.

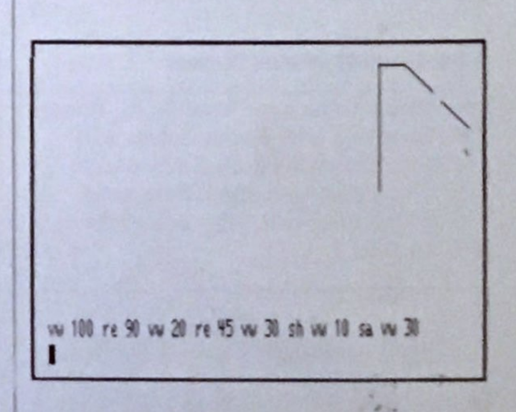


Bild 2. Eine einfache Befehlssequenz in GOSI

Bild 2 zeigt ein Programmbeispiel; ein Beispiel zum Übersetzungsvorgang. Man könnte folgendes GOSI-Programm:

VW 20 RE 40 LI 50 RW 10

in den nachfolgenden Code umsetzen:

```
21 #20.W
CD SCHREITE
21 -#40.W
CD DREHE
21 #50.W
CD DREHE
21 -#10.W
CD SCHREITE
```

Wichtig ist, daß man in GOSI auch neue Befehle definieren kann. Das geschieht mit dem Befehl:

LERNE

Danach folgt der Name des zu lernenden Programms. Danach die Befehle des Programmes bis zum Befehl ENDE, der angibt, daß hier die Definition endet.

Als Konstruktion für Schleifen gibt es den Befehl:

WIEDERHOLE oder einfach WH

Danach folgt die Anzahl der Wiederholungen und dann in eckigen Klammern die Befehle, die wiederholt werden sollen. So zeigt *Bild 3* die Definition eines Quadrats und *Bild 4* die Definition eines Kreises.

```
lerne quadrat
wh 4[vw 100 re 90]
ende
```

Bild 3. Ein Quadrat wird definiert

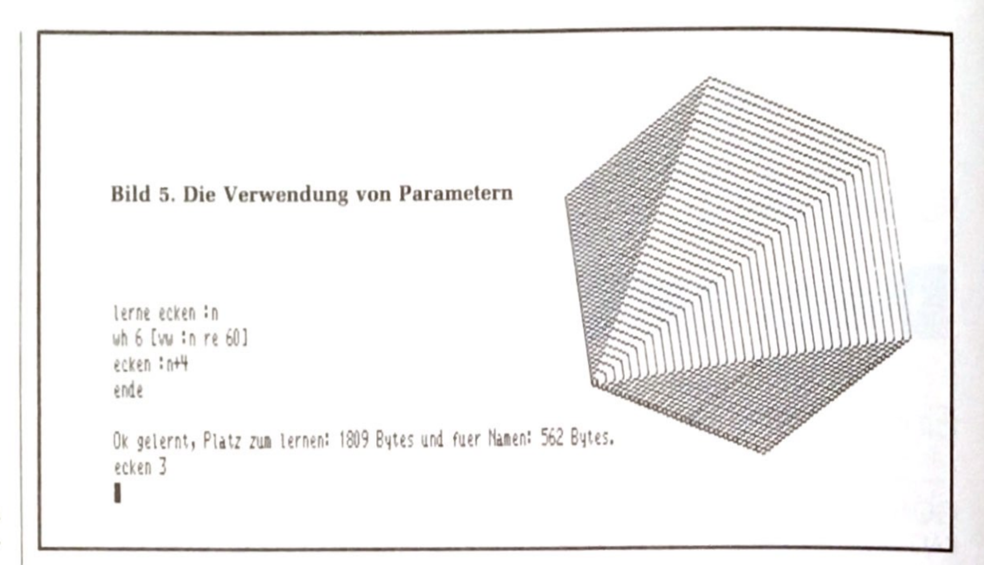
Jetzt kommt etwas Neues

Die Verwendung von Parametern. Wenn man Quadrate oder Kreise haben will, die unterschiedlich groß sind, so muß man dem Programm einen Parameter, die Größen mitgeben. Dies geschieht z. B. in *Bild 5*.

```
lerne kreis
wh 36 [vw 10 re 10]
ende
```

Bild 4. Ein Kreis wird definiert

Der Wert „:n" in der Zeile mit dem Lerne-Befehl definiert einen Parameter mit dem Namen „n". Der Doppelpunkt wird als Erkennung benutzt. In dem Programm kann man den Parameter „n" so verwenden, als sei es eine Zahl. Wenn man dann das Programm aufruft, so muß man eine Zahl hinter dem Befehl mit angeben.


Bild 5. Die Verwendung von Parametern

Beispiel: Ecken 3

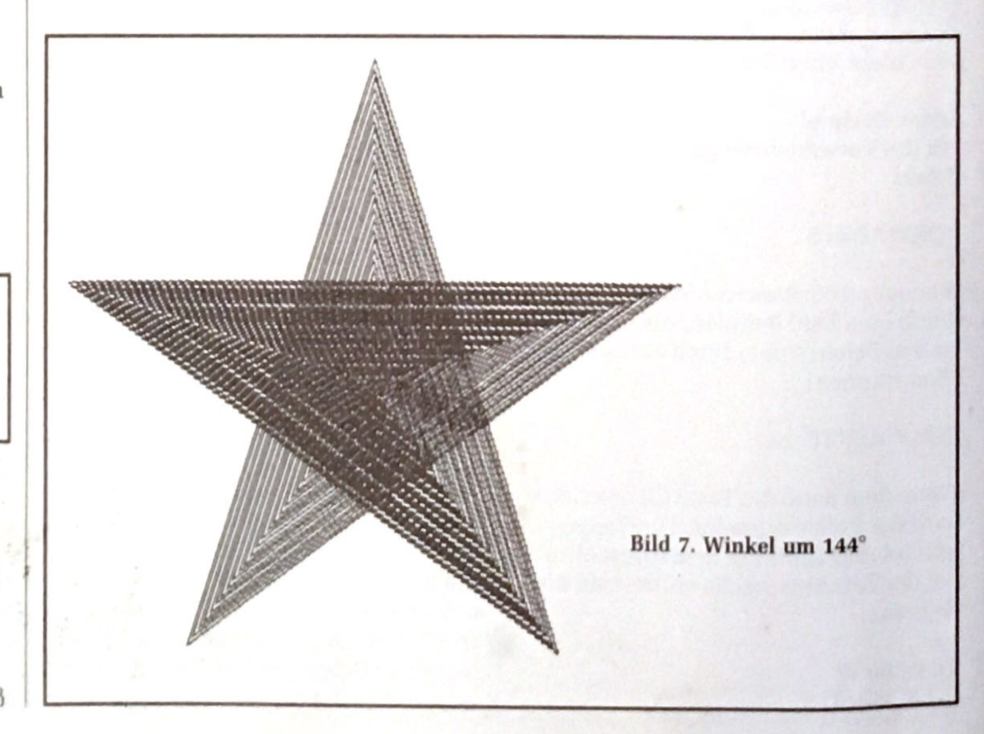
Die Zahl 3 wird dann dem Parameter „n" zugeordnet und steht in dem Programm als solcher zur Verfügung. In dem Programm ECKEN geschieht aber noch was anderes. Am Schluß steht der Befehl „ecken :n+4". Das bedeutet, daß Programm ruft sich selbst erneut auf, verwendet aber diesmal als Parameter den um vier erhöhten Wert von „n" beim nächsten Durchlauf.

```
lerne vspirale :laenge :winkel
vw :laenge re :winkel
vspirale (:laenge+3) :winkel
ende
```

Bild 6. VSPIRALE – Ein Programm mit vielen Gesichtern

Dieses Programm wird nie enden. Man nennt diese spezielle Aufrufform auch „tail recursion" in der Wirkung kommt sie einem Sprung gleich, nur daß auch Parameter verwendet werden.

Wenn man das Programm anhalten will, so muß man die Tasten „CTRL" und gleichzeitig „S" drücken. Will man das Programm weiterlaufen lassen, so drückt man „CTRL" und gleichzeitig „Q". Will man das Programm abbrechen, so muß man zuerst „CTRL" und „S" gedrückt haben und dann „CTRL" und „C" drücken.


Bild 7. Winkel um 144°

Man kann auch mehrere Parameter angeben, wie in *Bild 6*. Es ergeben sich dann Figuren, wie *Bild 7*, *Bild 8*, *Bild 9* und *Bild 10*. Jeder dieser Bilder zeigt eine besondere Form des Programms „VSPIRALE". Man kann es einmal mit ähnlichen Winkeln beim Aufruf probieren und wird diese Grundtypen immer wieder erkennen.

Da GOSI sehr umfangreich ist, sei hier auf die dem EPROM-Satz beigefügte Beschreibung sowie auf das Buch „Einführung in LOGO" von Harald Abelson, erschienen im IWT-Verlag, München, verwiesen, in dem es viele ausführliche Beispiele gibt, die den Umfang hier sprengen würden. Abschließend noch ein paar besondere Anwendungen von GOSI, die in den genannten Werken nicht vorkommen.

```
lerne lichtgriffelxy
solange (:port 240)&1<>0 [seite 0 0]
port 112 8
solange (:port 112)&1=0 [seite 0 0]
setze "x (:port 124)*2  setze "y (:port 125)*2
ende
```

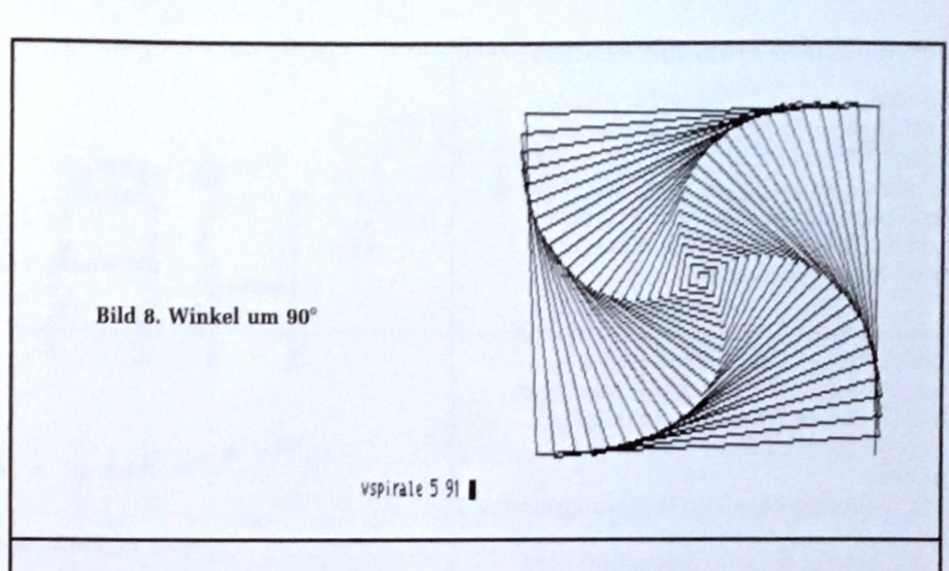
Bild 11. Lichtgriffel-Programm

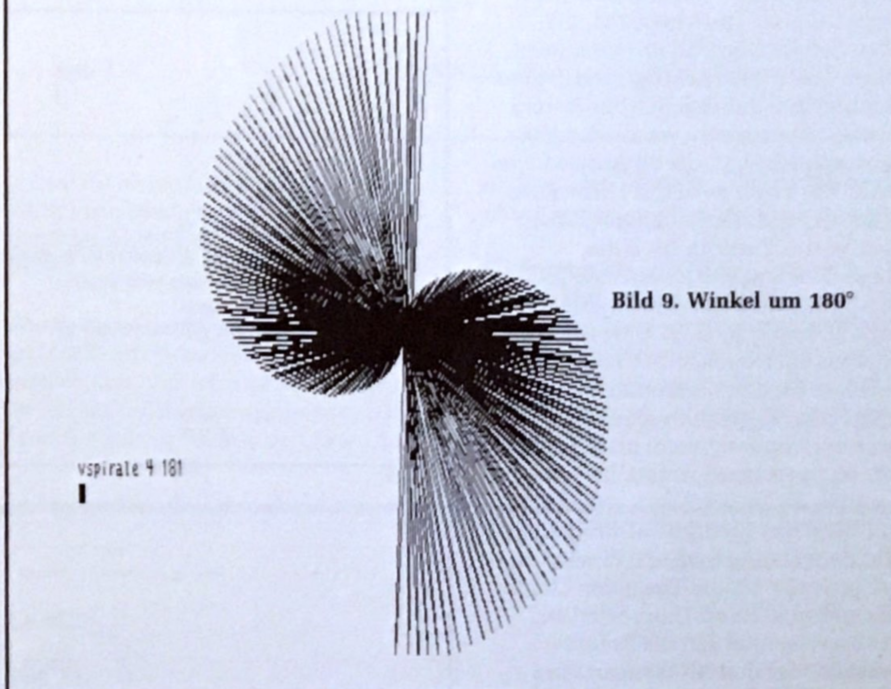
```
lerne testgriffel
lichtgriffelxy
dr :x dr [ ] dz :y
testgriffel
ende
```

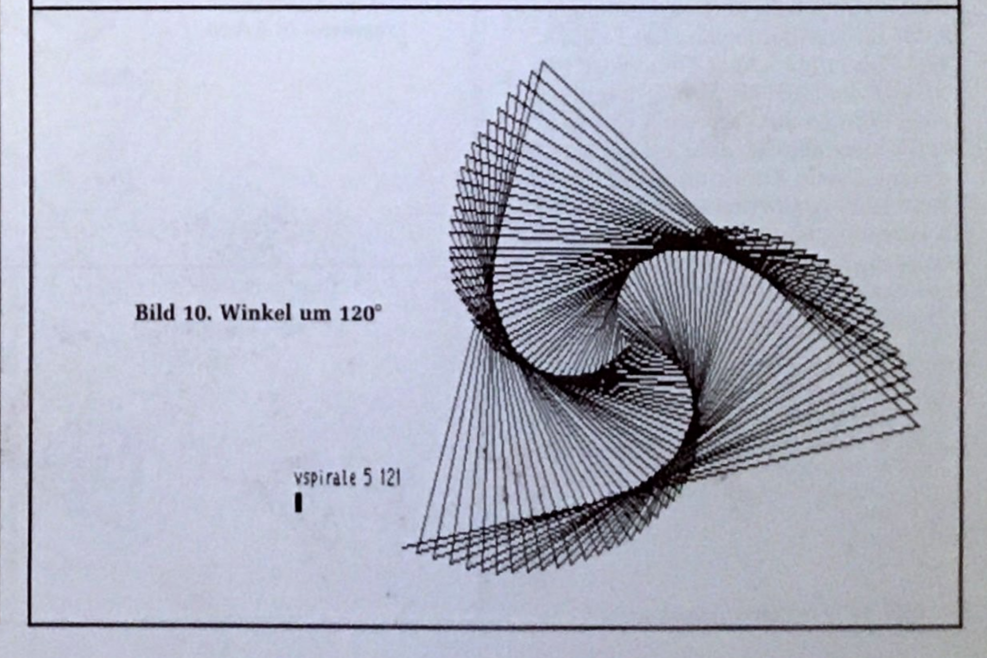
Bild 12. Testprogramm

```
88 142
96 182
120 182
184 201
240 201
336 192
432 96
```

Bild 13. Testausgabe


Bild 8. Winkel um 90°


Bild 9. Winkel um 180°


Bild 10. Winkel um 120°

Der Anschluß eines Lichtgriffels

Die Baugruppe GDP64 ist für den Anschluß eines Lichtgriffels vorbereitet. Hier ein Beispiel, wie man den Lichtgriffel mit GOSI bedienen kann. GOSI eignet sich nämlich nicht nur zur Erstellung von Grafiken, sondern auch hervorragend für Steuerungen.

Der Lichtgriffel wird an den Anschluß LPCK, PIN 21 des EF 9366 angeschlossen. Dort befindet sich ein Widerstand 330 Ω, der nach Masse geschaltet ist. Den Widerstand kann man dann entfernen.
Der Lichtgriffel muß im Normalzustand 0V führen und einen kurzen Puls nach +5 V liefern, wenn er beleuchtet wird. Ferner wird eine Taste benötigt, die meist in den Lichtgriffel mit eingebaut ist. Diese Taste wird betätigt, wenn man den Lichtgriffel auf den Schirm drückt. Die Taste soll angeben, wann man die Position erreicht hat, die angemerkt werden soll. Die Taste wird mit Hilfe eines Ports auf der IOE-Karte angeschlossen. Dazu wird die Taste an Bit 0 des 74LS245 (B1) angeschlossen. Die IOE-Karte wird auf die Adresse Fx gelegt, also alle Brücken (4, 5, 6, 7) werden offen gelassen. Der 74LS245 ist dann auf der Adresse F0 oder dezimal 240 erreichbar. Man kann auch eine andere Adresse verwenden, wenn man das Programm entsprechend umstellt.

Bild 11 zeigt das Lichtgriffel-Programm. Mit der Anweisung SOLANGE wird zunächst gewartet bis die Taste des Lichtgriffels gedrückt wird. Dann wird auf Port 112 (sedezimal 70) der Befehl 8 ausgegeben, der den Bildschirm kurz weiß leuchten läßt, und zwar solange, bis der Lichtgriffeleingang des EF9366 einen Puls entdeckt hat. Dann wird mit den Befehlen ‚SETZE "X (....)' der eingelesene Wert an die Variable X übergeben. Die Variable "X steht auf der linken Seite mit einem Anführungszeichen, da ihr ein Wert zugewiesen wird, während sie rechts mit einem Doppelpunkt versehen ist, da ein Wert aus der Variablen genommen wird. Die Variable "Y wird entsprechend belegt.

```
lerne zeichne
lichtgriffelxy
stifthoch aufxy :x :y stiftab
wiederhole 6 [vorwaerts 4 rechts 60]
zeichne
ende ▌
```

Bild 14. Das Zeichne-Programm

Bild 15. Zeichnungen mit dem Lichtgriffel

```
lerne menue
bild blinker 0 0 dz [Quadrat :0] blinker 0 4 dz [Kreis   :0]
lichtgriffelxy  wenn :y<60 [kreis]
wenn :y>60 [quadrat]
solange (:port 240)&1=0 [seite 0 0]
menue
ende
```

Bild 16. Programm zur Menüeingabe

Quadrat :0

Kreis   :0

Bild 17. Das Menü-Programm in Arbeit

```
lerne achteck
wh 8 [vw 50 re 45]
ende ▌
```

Bild 18. Das ACHTECK als Bewegungsmuster

Bild 12 zeigt ein kleines Testprogramm, mit dem man die Eingabe prüfen kann. Es liefert als Ergebnis zum Beispiel *Bild 13*, die Koordinaten, wenn man den Lichtgriffel auf den Bildschirm drückt und damit die Taste betätigt.
Nun kann man ein Zeichenprogramm schreiben. *Bild 14* zeigt ein Beispiel. Ein kleines Sechseck wird an der Position ausgegeben, bei der man die Taste am Lichtgriffel gedrückt hat. *Bild 15* zeigt eine damit erstellte Zeichnung. Aufgrund der Arbeitsweise des EF9366 kann man in horizontaler Richtung aber nur jeden achten Punkt erreichen, während in der vertikalen Richtung jeder Punkt erreichbar ist.

Auch eine Menüsteuerung ist möglich. *Bild 16* zeigt ein Beispiel. Man tippt mit dem Lichtgriffel einen Menüpunkt an und erhält entweder KREIS oder QUADRAT. Die beiden Programme müssen zuvor natürlich auch eingegeben worden sein. *Bild 17* zeigt das Ergebnis bei KREIS. Der KREIS oder das QUADRAT erscheinen dabei solange, bis die Taste am Lichtgriffel wieder losgelassen wird.

Bewegte Grafik

Man kann auch Bilder über den Bildschirm bewegen. *Bild 18* zeigt das Programm für ein Achteck, das über den Bildschirm bewegt werden soll. In *Bild 19* ist das BEWEGE-Programm abgebildet.

Um die Bewegung störungsfrei zu gestalten, wird eine besondere Technik verwendet. Das Bild wird zunächst unsichtbar auf einer Bildebene gezeichnet. Erst danach wird die Bildebene umgeschaltet. Vor dem Zeichnen wird eine eventuell noch vorhandene alte Figur gelöscht.

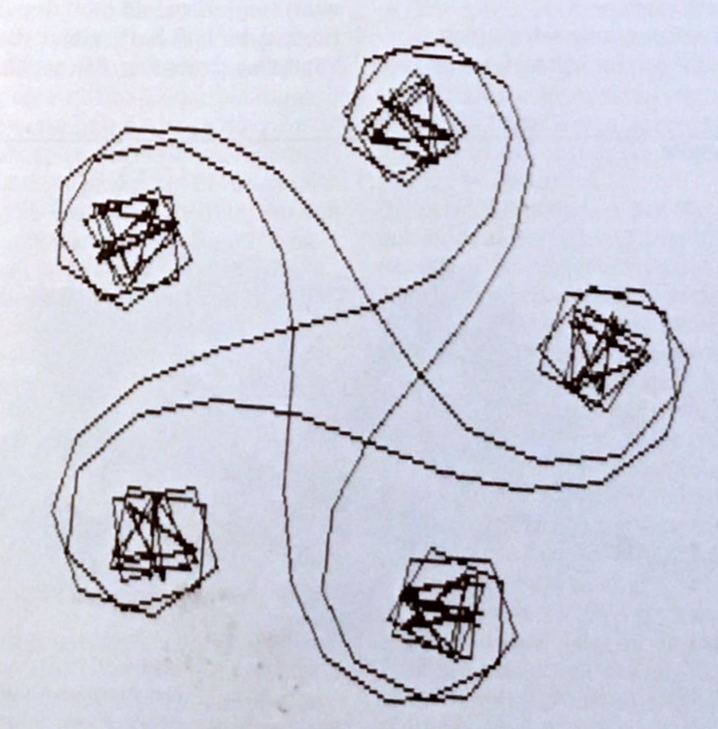
Als Parameter wird dem Unterprogramm die Anzahl der Bildpunkte zwischen zwei Bewegungsvorgängen gegeben. Damit läßt sich die Geschwindigkeit der Bewegung steuern. *Bild 20* zeigt einen Ausschnitt aus der Bewegungsphase.

Bild 19. Das BEWEGE-Programm

```
lerne bewege :speed
vi seite 1 0 stift 1 achteck stift 0 re 90 sh vw :speed*2 sa li 90 achteck
seite 0 1 stift 1 li 90 sh vw :speed sa re 90 achteck stift 0
re 90 sh vw :speed*2 sa li 90 achteck li 90 sh vw :speed sa re 90
bewege :speed
ende ▌
```

```
bewege 3
▌
```

Bild 20. Ausschnitt aus der Bewegung



```
BEISPIEL 18:

8800:
  KNAEUL:=$
  21 #1.W
  22 8900.W
  21 #180.W
  CD SCHLEIFE
  21 #40.W
  CD SCHREITE
  2A 8900.W
  CD DREHE
  2A 8900.W
  11 #10.W
  19
  22 8900.W
  CD ENDSCHLEIFE
  C9
```

R. Schulé

Roboter, Automaten und Grafikgeräte aus dem Baukasten

Der Fischertechnik-Baukasten

In der NDR-Fernsehserie „Einführung in die Mikroelektronik" geht es in erster Linie um den Aufbau und die Programmierung des NDR-Klein-Computers. Dennoch sollen Anwendungen nicht zu kurz kommen. Um die Kosten niedrig zu halten und einen problemlosen Aufbau zu ermöglichen, bietet Fischertechnik zu dem NDR-Klein-Computersystem einen Bausatz für sechs verschiedene Modelle an.

Historisch gesehen haben sich Computer aus den Aufgabenstellungen der numerischen Kalkulationen in technisch-wissenschaftlichen Anwendungen entwikkelt. Jüngere Anwendungsbereiche sind die Datei- und Textverarbeitung und die Steuerungstechnik. Während Kalkulationen nahezu alle Computer von Haus aus beherrschen und Datei- und Textverarbeitung sich heute schon auf vielen der kleinen Heim- und Selbstbaucomputer realisieren lassen, wirft die Steuerungstechnik neue Probleme auf. Derjenige, der nicht auf dem Gebiet der Steuerungstechnik arbeitet, sondern sich aus privatem Interesse oder im Verlauf seiner Ausbildung mit der Steuerungstechnik beschäftigt, kann sich in aller Regel nicht eine ausgewachsene Maschine oder gar eine Industrieanlage als Objekt seiner Computerexperimente hinstellen, er wird Modelle benutzen.
Der Aufbau dieser Modelle darf nicht übermäßig hohe Anforderungen an die handwerkliche Geschicklichkeit und die Werkstattausrüstung des Computeristen stellen. Dies bedeutet, daß die Modelle vereinfacht werden müssen, ohne jedoch die Charakteristika der simulierten Anlage aufzugeben.
Das Fischertechnik-Konstruktionssystem erlaubt den funktionsrichtigen Aufbau von technischen Modellen aller Art. Die mit einer Genauigkeit von bis zu $^{5}/_{100}$ mm gefertigten Einzelteile sind sehr vielseitig verwendbar und gestatten Modell-Lösungen aus allen Bereichen der Technik. Diese Variationsbreite hat nicht nur Kinder und Jugendliche zu diesem System greifen lassen, auch Ingenieure in Industrielabors setzen Fischertechnik zur wirklichkeitsnahen Simulation ein. Selbst auf internationalen Messen fällt dieses patente Konstruktionssystem immer wieder bei der Verdeutlichung technischer Vorgänge auf.

Der Baukasten „Computing"

Die Modelle im Baukasten Computing wurden auf Anregungen des Teams der NDR-Fernsehserie „Mikroelektronik" entwickelt. Wer sich das Sortiment an Bauelementen in dem Fischertechnik-Kasten anschaut, wird sofort herausfinden, daß außer dem Roboter, dem eine der Folgen gewidmet ist, sich noch eine Menge mehr machen läßt. Einige der Ideen wurden von den Fischertechnik-Entwicklern aufgegriffen und sind in dem Begleitheft des Baukastens dargestellt. Andere Modelle ergeben sich sicherlich, wenn Sie Ihre Phantasie spielen lassen. Wer in einer Ampelanlage nicht nur die drei verschiedenfarbigen Leuchtdioden sehen will, dem sei zum Beispiel gleich empfohlen, in den Baukasten zu greifen und die Teile für einen Ampelmast mitsamt den drei Lampen herauszuholen. Wir würden dann aber auch empfehlen, auf dem Experimentierfeld der IOE-Karte gleich die Verstärkerstufe aufzubauen, die nachher auch für die weiteren Modelle verwendet werden kann. Denn die Strombelastbarkeit der Original-Ampelschaltung reicht nicht für die Glühlampen aus. Relais, die mit einem weiteren Ausgabebit angesteuert werden können, erlauben auch die Umpolung des Stromflusses. Dies wird zwar bei der Verwendung von Lampen nicht benötigt, jedoch läßt sich auf diese Weise später der Drehsinn von Motoren umkehren.
Die weiteren Modelle seien nicht in der Reihenfolge der NDR-Sendung besprochen, sondern nach dem Schwierigkeitsgrad des Aufbaus und der Komplexität des Steuerungsprogramms. Da wird als erstes die Sortieranlage aus dem Baukasten stehen (*Bild 1*). Sie stellt eine Variation der Aufgabe „Gut und Schlecht" aus der Serie „Mikroelektronik" dar. Die Apparatur kann die zwei im Fischertechnik-System am häufigsten vorkommenden Bausteine verschiedener Länge unterscheiden.

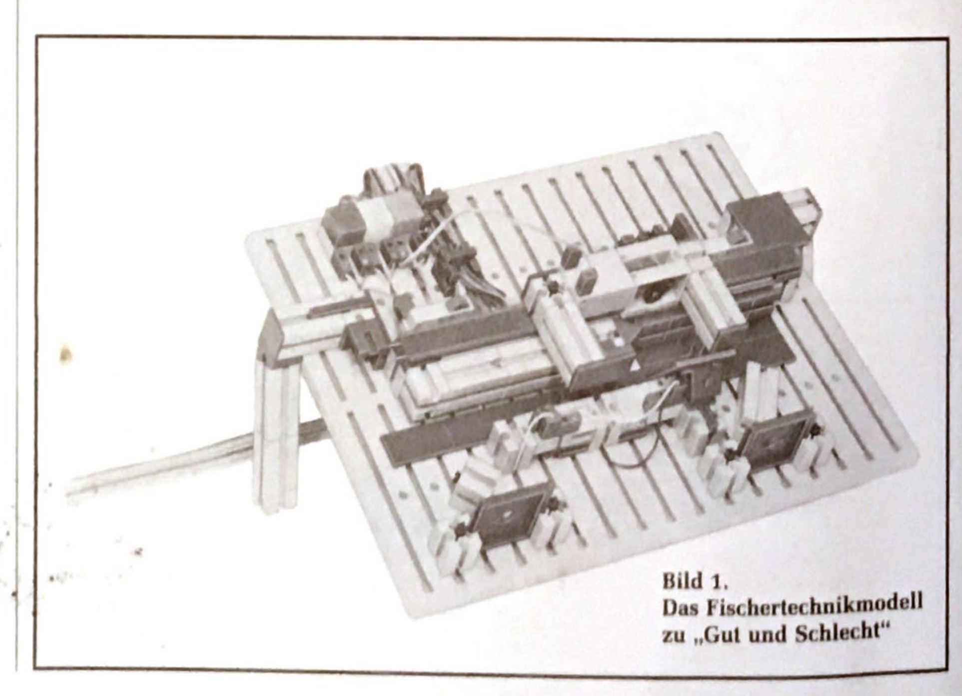


Bild 1. Das Fischertechnikmodell zu „Gut und Schlecht"

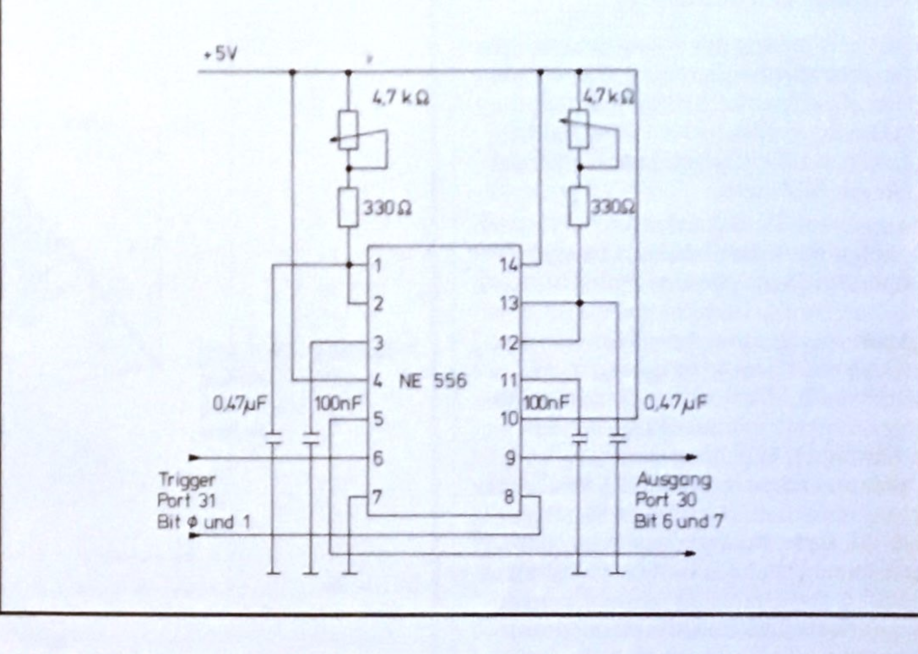


Bild 2. Diese Schaltung setzt die Potentiometerstellung linear in eine Pulslänge um

Die Sortieranlage

Auszug aus der Bauanleitung: „Der Baustein wird über eine Rutsche in die Meßstrecke befördert. Anschließend wird er von Hand nach links in den Meßkanal geschoben, wobei zwangsläufig der Starttaster betätigt wird. Messen heißt vergleichen, hier vergleichen wir die unbekannte Länge des Bausteins mit dem Abstand zwischen dem Starttaster und dem Meßtaster. Werden beide Mini-Taster gleichzeitig durch den Baustein betätigt, so muß es sich um den längeren Baustein 30 handeln. Der Baustein 15 kann nur einen Mini-Taster betätigen. Nachdem erkannt und durch die Lampen angezeigt ist, um welchen Baustein es sich handelt, erfolgt der Sortierschritt. In einem Fall läuft der Schlitten, der den Baustein schon in den Meßkanal transportiert hat, weiter und wirft den Baustein am linken Ende aus. Im anderen Fall wird der Motor des Schlittens umgepolt, so daß der Baustein nach rechts mitgenommen wird. Drei Taster, die längs der Bahn des Schlittens angeordnet sind, melden dem Computer das Erreichen der Endlagen und der Ausgangsstellung in der Mitte, zu der der Schlitten zuletzt wieder zurückkehrt."

Wer schnell ist, muß warten

Wer die Sortieranlage programmiert hat, wird schon ein Gespür für die Programmierung mechanischer Apparaturen haben. Was immer wieder im Vordergrund der Echtzeitprogrammierung steht, ist die Synchronisation der Abläufe im Computer und dem Modell. Wem diese Thematik neu ist, der sollte zunächst davon ausgehen, daß der Computer immer schneller als die angeschlossene Mechanik ist und daher die Aufgabe bekommt, sich in eine Warteschleife zu begeben. Dabei müssen in jedem Durchlauf die Eingänge gelesen werden und das Erreichen des Zielzustandes (z. B. Wagen in der Mitte positioniert) überprüft werden. Bei Erreichen des Ziels wird die nächste Aktion auf der Ausgabeseite ausgelöst (z. B. Motor abschalten). Erfahrene Programmierer werden durch die Taster einen Interrupt auslösen, um so Zeit für ein Hintergrundprogramm zu gewinnen.
Beide Verfahren haben den Nachteil, daß an den Computern keine Vorwarnungen kurz vor dem Erreichen des Ziels abgegeben werden. Aus diesem Grund enthält der Baukasten zwei Potentiometer, die der stufenlosen Positionsmeldung dienen. Je nach Modell sind die Potentiometer in dem Zentrum eines Drehkranzes eingebaut oder werden über ein Zahnrad angetrieben.
Wenn in dem Computer eine Eingabemöglichkeit für Analogwerte existiert, kann eine wesentlich verbesserte Positionsmeldung erfolgen. Für den Analogeingang gibt es verschiedene Schaltungsvorschläge. Hier im Heft ist eine Schaltung mit einem spannungsgesteuerten Frequenzgenerator angegeben. Wird das Potentiometer als Spannungsteiler geschaltet, so kann an dem Schleifer eine der Position proportionale Spannung abgenommen werden. Das IC 74LS627 erzeugt dann im nächsten Schritt eine der Position proportionale Frequenz. Benutzt man nun den Computer, um eine halbe Schwingungsperiode der Rechteckschwingung auszuwählen, so liegen die Positionsdaten in dem Computer zur Weiterverarbeitung vor. Das EPROM Roboter geht von dieser Schaltung aus. Wenn Sie dieses EPROM verwenden wollen, sollten Sie die Analogeingabe so aufbauen.

Nochmals: A/D-Wandlung

Wir wollen jedoch einen kleinen Nachteil dieser Schaltung nicht verschweigen. Bei dem geschilderten Auszählverfahren ist der Zählwert der Impulsbreite proportional und damit reziprok zur Frequenz und letztlich zur Position des Potentiometers. Wird eine lineare Ablesung gefordert, muß die Berechnung des Reziprokwertes erfolgen. Die nachfolgend geschilderte Schaltung nach [1] (siehe *Bild 2*) vermeidet diesen Nachteil. Sie benutzt das Potentiometer als zeitbestimmendes Element in einer Monoflop-Schaltung. Die Impulsdauer des Monoflops, die wieder genauso ausgezählt wird, ist der Potentiometer-Schaltung direkt proportional. Allerdings muß das Monoflop durch einen Impuls des Computers gestartet werden. Bei der reichlichen Ausstattung der IOE-Platine stellt dies jedoch kein Problem dar.

Achtung: Trigonometrie

Die Verwendung der Potentiometer läßt sich sehr anschaulich mit Hilfe des Modells „Grafiktablett" (*Bild 3*) aus dem Baukasten verdeutlichen. Ein Grafiktablett ist ein Eingabegerät zur Konstruktion am Bildschirm.

Auszug aus der Bauanleitung: „Wir verwenden hier eine einfache Lösung mit einem Knickarm, dessen Endpunkt über die Zeichenfläche geführt wird (*Bild 4*). In den Angelpunkten des Arms sind jeweils Potentiometer angebracht. Anhand von Bild 4 wollen wir uns die geometrischen Zusammenhänge veranschaulichen. Zur Abspeicherung im Computer und zur Erstellung von Zeichnungen auf dem Bildschirm benötigen wir die Koordination des Bildpunktes in waagerechter und senkrechter Richtung: x und y. Der Ursprung dieses sogenannten kartesischen Koordinatensystems liege in der linken unteren Ecke der Zeichenfläche.

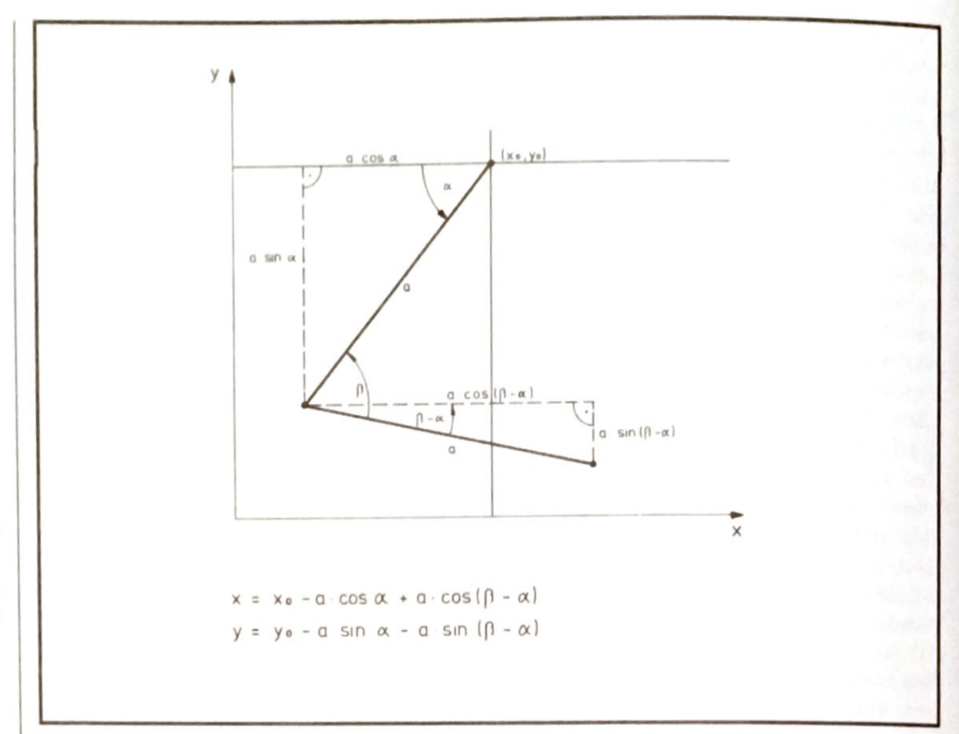


Bild 4. Das ist die Geometrie, die hinter dem Grafiktablett steht

Mit (x_0, y_0) haben wir die Position des ersten Drehpunktes, nämlich den Mittelpunkt des Drehkranzes, bezeichnet. Von da aus folgen wir in Gedanken der ersten Hälfte des Armes und kommen zu dem zweiten Drehpunkt. Wer die Grundkenntnisse der Trigonometrie parat hat, erkennt leicht, daß die neuen Koordinaten um die Achsabschnitte $a \cdot \cos\alpha$ und $a \cdot \sin\alpha$ verschoben liegen. Dabei bezeichnet a die Länge des „Oberarms" und α den Winkel des Potentiometers.

Weiter geht es entlang des „Unterarms" zum Griffel. Damit die Berechnung vereinfacht wird, hat dieses Teilstück gleichfalls die Länge a. Wohl dem, der sich der Schulmathematik über Winkel und Parallelen und Geraden erinnert. Er wird erkennen, daß der Winkel zur Waagrechten sich aus der Differenz der beiden Potentiometerwinkel α und β ergibt. Nehmen wir auch noch die zu diesem Winkel gehörigen Achsabschnitte hinzu, so erhalten wir die Endformeln, wie sie in Bild 4 angegeben sind.

Bei der Programmierung berücksichtigen wir noch, daß der Taster am rechten Rand der Zeichenfläche verwendet wird, um dem Computer mitzuteilen, wann der Griffel positioniert ist. Die weiteren Taster stehen für sonstige Funktionen, wie „Bildschirm löschen", „Seite umschalten" usw. zur Verfügung. Sie können aber auch mit einfachen geometrischen Figuren hinterlegt sein, die nur wenige Punkte zu ihrer Konstruktion benötigen: „Zeichne Strecke", „Zeichne Dreieck", „Zeichne Kreis".

Wenn wir alles zusammenfassen, was wir uns bisher erarbeitet haben, die Steuerung von Motoren, das Einlesen von Tastern und Potentiometern, so haben wir die Hilfsmittel zur Programmierung von Robotern vollständig. Wer sich langsam an die Materie heranarbeiten möchte, dem empfehlen wir, zunächst den Vertikalroboter (*Bild 5*) aufzubauen. Mit dieser Maschine können verschiedene Aufgaben programmiert werden.

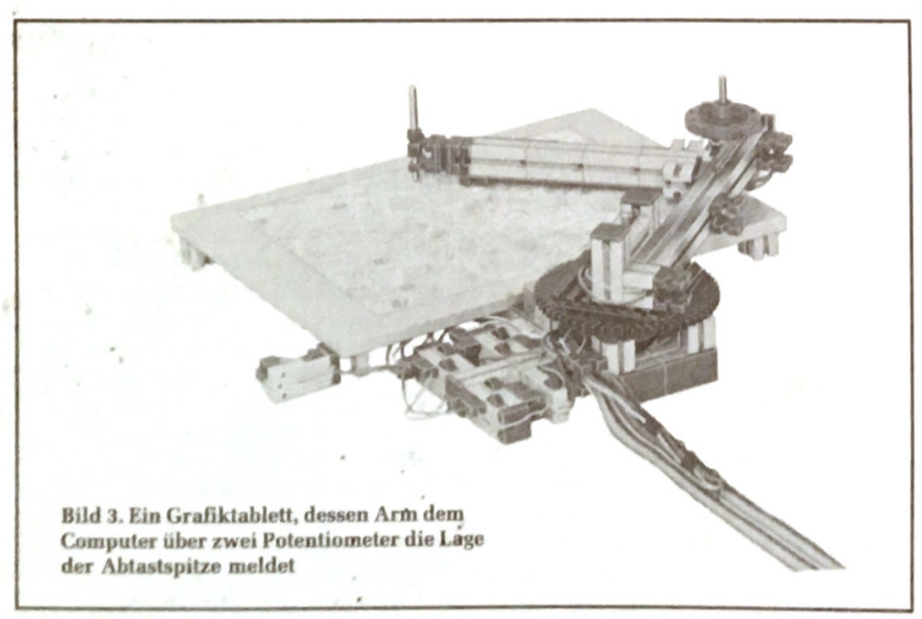


Bild 3. Ein Grafiktablett, dessen Arm dem Computer über zwei Potentiometer die Lage der Abtastspitze meldet

Die Türme von Hanoi

Die Problemstellung geht auf die Überlieferung zurück, nach der buddhistische Mönche mit einer Aufgabe beschäftigt sind, die ein außerordentliches Maß an Geduld erfordert:

Auf einem Brett stehen drei Pfähle, aus Kupfer, Silber und Gold. Irgendwann vor langen Zeiten staken auf dem Kupferstab hundert Lochscheiben, alle mit verschiedenem Durchmesser und der Größe nach geordnet. Die Aufgabe ist nun, den Stapel auf den goldenen Pfahl umzuschichten. Dabei gelten folgende Regeln:

1. Es darf jeweils nur eine Scheibe bewegt werden.

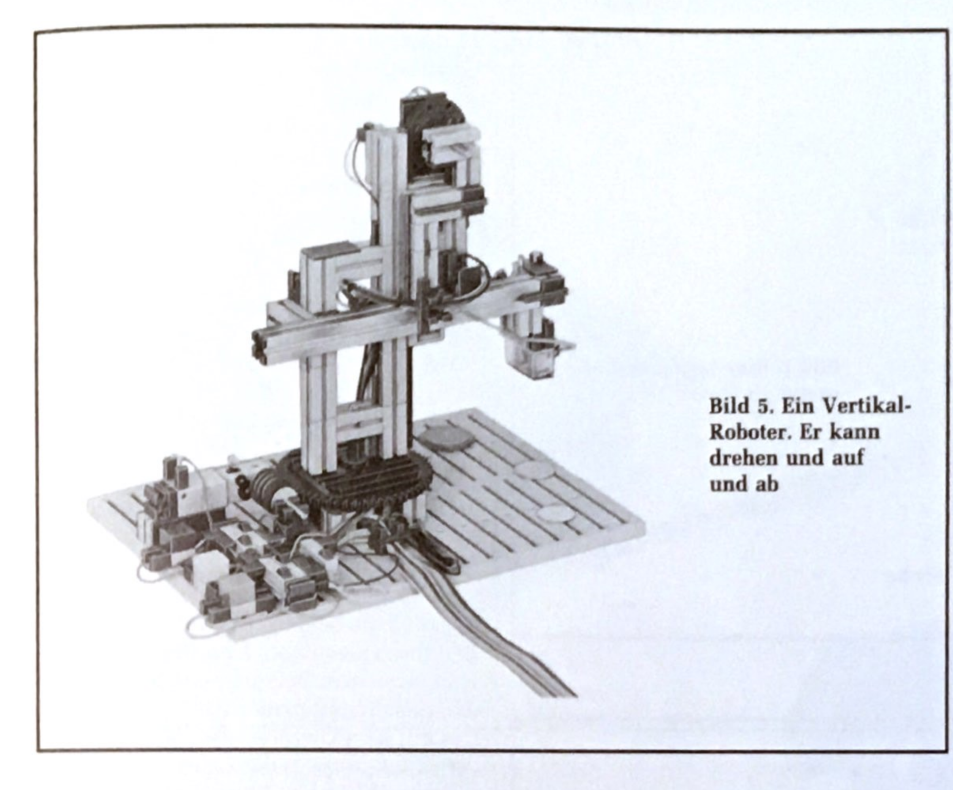


Bild 5. Ein Vertikal-Roboter. Er kann drehen und auf und ab

2. Es darf nie eine größere Scheibe auf eine kleinere gelegt werden.
3. Zur Ablage dürfen nur die drei Pfähle benutzt werden.

Die Überlieferung berichtet weiter, daß bei Erreichung des Ziels die Welt untergeht.

Diese Aufgabe eignet sich hervorragend zur Programmierung und kann von dem Roboter mit fünf Scheiben leicht ausgeführt werden. Der Algorithmus ist in folgendem kurzen rekursiven Basic-Programm angegeben (Bild 6):

Die eigentliche Steuerung des Roboters wird als Unterprogramm anstelle oder zusätzlich zu den Zeilen, die den Ausdruck bewirken, eingefügt.

Die Steuerung des Roboters kann relativ einfach gehalten werden, wenn man berücksichtigt, daß am dem oberen Wegende der Vertikalachse ein Endschalter sitzt. Bei der Abwärtsbewegung spricht ebenfalls ein Endschalter an, wenn der als Greifhand dienende Elektromagnet auf einen Widerstand stößt. Es ist somit nicht unbedingt notwendig, diese Bewegung mit dem Potentiometer abzufragen. Damit verbleibt die Drehung um die Vertikalachse als einzige stufenlos einstellbare Achse, was die Programmierung vereinfacht.

Wie Experimente gezeigt haben, kann eine solche Steuerungsaufgabe durchaus noch in Basic gelöst werden. Obwohl die Arbeitsgeschwindigkeit eines jeden Basic-Interpreters um Größenordnungen hinter der von Maschinensprachprogrammen zurücksteht, werden mit folgendem Basic-Programm recht gute Resultate erzielt:

```
1200 REM LAGEREGELUNGSROUTINE
1210 D=USR(EX)–SOLLWERT
1220 IF D>0 THEN SYS TURM,LINKS
1230 IF D<0 THEN SYS
     TURM,RECHTS
1240 IF D=0 THEN RETURN
1250 D=ABS(D) : IF D>5 THEN 1210
1260 FOR I=0 TO D
1270 NEXT
1280 SYS TURM,AUS
1290 GOTO1210
```

Hiermit möchten wir den Heimcomputerbesitzer ermutigen, sich auch mit seinem Basic-Computer an die Steuerungsthematik heranzuwagen.

Das kurze Programm zeigt auch, wie sich die Geschwindigkeit eines Gleichstrommotors durch das Tastverhältnis ein/aus variieren läßt. Gleichzeitig dürfte bei genauem Hinschauen auch das Geschwindigkeitslimit von Basic erkennbar sein. Der Motor läuft infolge der Pulsbreitenmodulation schon etwas unrund. Optimal wäre eine feste Frequenz von 20 bis 50 Hz, mit der die Stromversorgung des Motors getaktet wird. Aufgrund der Ankerträgheit ruckt der Motor dann noch nicht. Zu hoch darf die Frequenz jedoch auch nicht sein, da sonst der Motoranker als Lautsprecher arbeitet und dabei langfristig durchaus die Motorlager zerstören kann. Mit Programmen in Maschinensprache läßt sich ein optimal abgestimmter Servoregelungskreis aufbauen.

Automaten mit zwei Achsen

Eine größere Herausforderung stellen die restlichen Modelle aus dem Bauka-

```
660 DIM X(4),Y(4),Z(4),S(3)
670 PRINT"[illegible]TURM VON HANOI"
680 INPUT"ZAHL DER SCHEIBEN";N
690 M=0
700 X(M)=N : Y(M)=1 : Z(M)=3
710 GOSUB 800
720 END
800 REM PSEUDOREKURSIVE ROUTINE
810 IF X(M)=0 THEN RETURN
820 M=M+1
830 X(M)=X(M-1)-1
840 Y(M)=Y(M-1)
850 Z(M)=6-Y(M-1)-Z(M-1)
860 GOSUB 800
870 M=M-1
880 PRINT "SCHEIBE";X(M);"VON SAEULE";Y(M);"NACH SAEULE";Z(M)
890 GOSUB 1000 : REM ROBOTER ROUTINE
900 M=M+1
910 X(M)=X(M-1)-1
920 Y(M)=6-Y(M-1)-Z(M-1)
930 Z(M)=Z(M-1)
940 GOSUB 800
950 M=M-1
960 RETURN
```

Bild 6. Der Algorithmus zu Turm von Hanoi in Basic

```
1200 REM LAGEREGELUNGS ROUTINE
1210 D=USR(EX)-SOLLWERT
1220 IF D>0 THEN SYS TURM,LINKS
1230 IF D<0 THEN SYS TURM,RECHTS
1240 IF D=0 THEN RETURN
1250 D=ABS(D) : IF D>5 THEN 1210
1260 FOR I=0 TO D
1270 NEXT
1280 SYS TURM,AUS
1290 GOTO1210
1300 REM ARM ABSENKEN
1310 SYS ARM,AB
1320 IF USR(UNTEN)=0 THEN 1320
1330 SYS ARM,AUS
1340 RETURN
1400 REM ARM ANHEBEN
1410 SYS ARM,HOCH
1420 IF USR(OBEN)=0 THEN 1420
1430 SYS ARM,AUS
1440 RETURN
2000 PRINTUSR(EX);USR(EY):GOTO2000
```

Bild 7. Eine Lageregelungsroutine in Basic

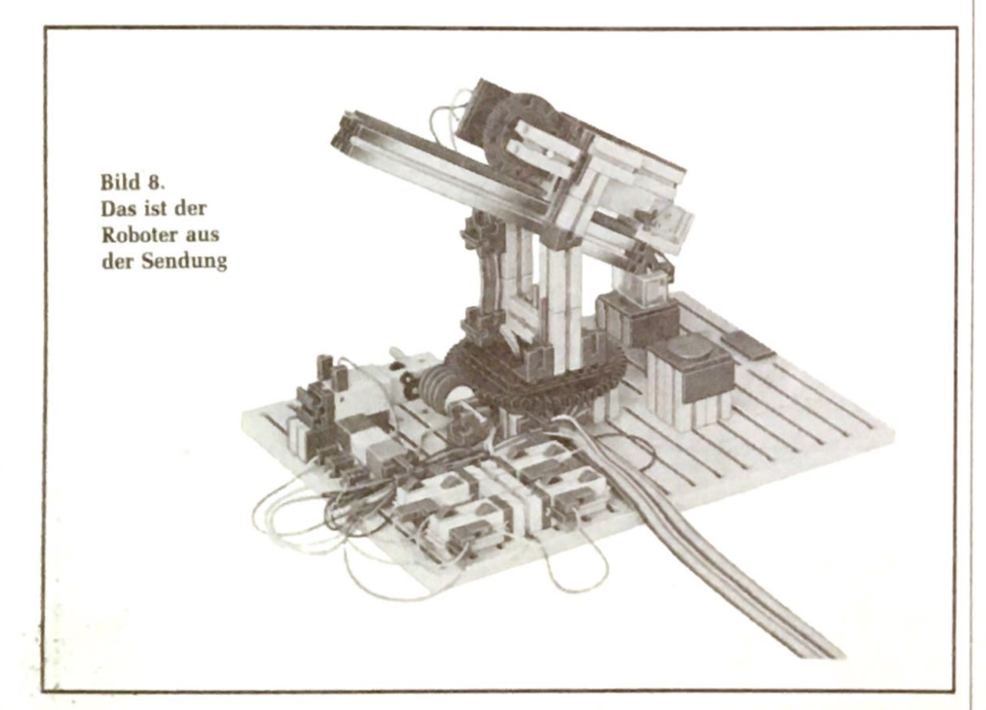


Bild 8. Das ist der Roboter aus der Sendung

sten dar. Sei es der zweite Roboter (*Bild 8*), der Plotter (*Bild 9*) und die Nachführanlage für Solarzellen (*Bild 10*). Letzteres Modell kann aufgrund der Aufhängung des Rahmens eine Solarzelle senkrecht zum Einfall der Sonnenstrahlen stellen, um so die einfallende Energie maximal zu nutzen. Wer dieses Steuerungsprogramm in Angriff nehmen will, sollte Kenntnisse der sphärischen Trigonometrie mitbringen, da recht komplexe Formeln den Sonnenstand als Funktion der geographischen Länge und Breite, der Uhrzeit und des Datums beschreiben. Eine Formulierung in Fortran ist allerdings auch veröffentlicht [2].

Alle genannten Modelle stellen zweiachsige Automaten dar, die in beiden Bewegungsachsen kontinuierlich eingestellt werden können. Bei diesen Modellen ist eine Strategie notwendig, in der die beiden Motoren angesteuert werden. Die naheliegende Lösung besteht in einem zeitlichen Nacheinander der Ansteuerungsroutinen. Dies führt jedoch zu eckigen Bewegungen. Zwar wird jemand, der Roboter in erster Linie aus Science-Fiction-Filmen kennt, dies als roboterhaft einschätzen. Jedoch zeigt die industrielle Praxis, daß man wegen der dabei auftretenden hohen Beschleunigungsmomente und dem damit verbundenen Energiebedarf an „glatten" Bewegungsabläufen interessiert ist. Verschiedene Verfahren können schon mit diesen einfachen Modellen studiert werden. Schematisch haben wir in *Bild 11* einige der Strategien aufgezeichnet. In Bild 11b wird die Nacheinanderausführung aus Bild 11*a* in eine Treppenkurve zerlegt. Sind die Stufen fein genug, so kann durchaus der Eindruck einer glatten Bewegung entstehen. Allerdings ist diese Bewegung nicht sehr schnell, da zu einem jeden Zeitpunkt immer nur ein Motor in Betrieb ist. In Bild 11c wird dem durch den gleichzeitigen Betrieb zweier Motoren Rechnung getragen. In aller Regel kann man nicht davon ausgehen, daß die beiden Motoren zu gleicher Zeit ihren jeweiligen Koordinatenendwert erreichen. Bei einer entsprechenden Geschwindigkeitsreduktion des Motors mit dem kürzeren Weg läßt sich erst der Knickpunkt in der Bahnkurve vermeiden (Bild 11d). Wer nun meint, mit der geraden Linie in dem Koordinatenfeld sei er an dem Ziel seiner Wünsche angelangt, sollte dies bei dem Plotter ausprobieren. Da in diesem Fall kein kartesisches, sondern ein Polarkoordinatensystem vorliegt, stellt sich die gerade Raumlinie als gekrümmte Linie in der Koordinatendarstellung dar (Bild 11e). Entsprechend muß dies bei der Programmierung der Motoransteuerung berücksichtigt werden. Dennoch können insbesondere in der Robotertechnik andere Überlegungen ein Abweichen von der geraden Raumkurve ratsam erscheinen lassen. Bedenkt man, daß der Roboter eventuell nicht unerhebliche Lasten transportieren muß, so ergeben sich mit Hilfe des entsprechenden mathematischen Apparats [3] Bahnkurven als Optimum zwischen Bewegungsdauer und Energieverbrauch wie in Bild 11*f* schematisch dargestellt.

Dieser kleine Exkurs sollte die Spannweite des Fischertechnik-Computing-Baukastens aufzeigen. Von dem Spaß an der Programmierung der hübschen kleinen Modelle bis zum ernsthaften Studium auf dem Gebiet der Robotik findet jeder, was er sucht.

Literatur

[1] *Goldkuhle, P., Kindt, V.:* Grafiktablett. mc 10/1983, S. 74.

[2] *Walraven, R.:* Calculating the Position of the Sun. Solar Energy Vol. 38/1978, S. 393.

[3] *Blume, C., Dillmann, R.:* Frei programmierbare Manipulatoren. Vogel-Verlag.

◀ Bild 9. Ein Plotter, der recht ordentlich zeichnen kann

Bild 10. Eine Anlage mit Nachführung für Solarzellen ▶

a b c d e f

Bild 11. Einige Wege, die zu verschiedenen Motoransteuermethoden gehören. Dabei handelt es sich um X- und Y-Motoren

Hans Hehl

Das Basic

8 KByte für den NDR-Klein-Computer

Puristen werden an dieser Stelle die Nase rümpfen, weil wir hier den Befehlssatz eines Basics besprechen. Da die Welt aber nun einmal nicht immer nach den Vorstellungen von Puristen geordnet ist, wollen wir hier nicht die Augen verschließen und so tun, als sei Basic nicht in der Welt. Heute kann jeder Schüler mehr oder weniger gut Programme in Basic schreiben und sie auf Basic-Computern ablaufen lassen. Vielleicht reizt es solche Kenner, auch die anderen Vorzüge des NDR-Klein-Systems kennenzulernen, wenn sie wissen, „der Computer kann auch Basic". Außerdem wird auch der völlig im Sinne der reinen Lehre erzogene Novize irgendwann auf Basic-Programmierer stoßen. Es ist dann gut, wenn er mitreden kann.

Das 8-KByte-Basic für den NDR-Klein-Computer entspricht in etwa dem Microsoft-Standard. Es wird in zwei EPROMs 2732A geliefert, die anstelle der beiden Grundsoftware-EPROMs auf der SBC-2-Platine eingesetzt werden. Da der Interpreter auch den Grafikprozessor steuern sollte, mußten aufgrund des knappen Speicherplatzes Abstriche am Bedienungskomfort gemacht werden. Sedezimalzahlen („Hexzahlen") werden in diesem Text durch ein nachgestelltes „H" gekennzeichnet. Basic-Befehle werden in der Beschreibung groß geschrieben, dürfen jedoch beliebig in Groß- oder Kleinbuchstaben eingegeben werden. Das Betätigen der Return-Taste wird durch <CR> angedeutet. <CTRL-Q> soll das Drücken der Control-Taste und das gleichzeitige Drücken der Q-Taste symbolisieren. Der Interpreter benötigt zusätzlich Speicherplatz bis zur Adresse 88C4H, ab 88C5H beginnt der Programmspeicher.

Die alphabetische Liste der Basic-Befehle

Die bei manchen Befehlen angegebene Klammer soll darauf hinweisen, daß dem Basic-Befehl ein Ausdruck in Klammern folgen muß, sonst erfolgt die Fehlermeldung „?SN Fehler" (SYNTAX ERROR), was die falsche Eingabe eines Basic-Befehls signalisiert.

Zum Start des Basic-Interpreters

Nach dem Einschalten des Rechners meldet sich der Interpreter mit einem Fragezeichen. Dann muß C (Großbuchstabe!, C für „Cold Sart") getippt werden, worauf folgende Meldung auf dem Bildschirm erscheint:

```
8K BASIC 1.3
RDK 83
o.k.
>
```

Wichtig: Nur „C" erweckt den Computer zum Leben! Wenn der Computer „schon im Laufen ist", dann können Sie nach Drücken der Reset-Taste (SBC-2-Platine) mit der Eingabe von „W" (für Warmstart) den Interpreter starten, ohne Ihr Programm im Speicher zu zerstören. Meldung: „o.k" und „>". Beim Warmstart umgeht der Interpreter die Initialisierungsroutinen.

Die Kontrollfunktionen

Während der Programmausführung fragt der Interpreter ständig die Tastatur ab. Bei Eingabe von <CTRL-S> stoppt die weitere Programmausführung sofort. Mit <CTRL-Q> wird dann wieder gestartet. Dies gilt auch für den Befehl LIST. Wird während der Programmausführung die Escape-Taste <ESC> betätigt, bewirkt dies eine Unterbrechung nach dem gerade ausgeführten Befehl sowie die Meldung „abgebrochen in Zeile XY". Wie auch bei dem Befehl STOP kann man jetzt die Variablen inspizieren, modifizieren und das Programm mit dem Befehl CONT fortsetzen, sofern es nicht verändert wurde.

Ein aktiver INPUT-Befehl kann nicht mit <ESC> unterbrochen werden, sondern nur durch die Betätigung der Return-Taste; der Interpreter geht daraufhin in die Basic-Anweisungsebene zurück (o.k.-Meldung), kann aber das Programm (per CONT) mit dem INPUT-Befehl fortsetzen, sofern es zwischenzeitlich nicht modifiziert worden ist.

Variablen

Basic läßt die Verarbeitung variabler Größen zu. Eine Rechenvorschrift kann zum Beispiel in Buchstaben-Form angegeben werden (c=a+b) und für die Variablen werden je nach Bedarf unterschiedliche aktuelle Werte eingesetzt.

Basic kann dabei nicht nur Zahlen als variable Größen verarbeiten, sondern auch Buchstaben und Texte (sogenannte „Strings"). Variablen für Zahlen werden durch eine maximal zweistellige, alphanumerische Zeichenkombination (alphanumerisch: aus Ziffern und Buchstaben zusammengesetzt) bezeichnet, wobei an erster Stelle immer ein Buchstabe stehen muß. String-Variablen (maximal zulässige Länge: 255 Zeichen bei entsprechender Speicherplatzreservierung) werden dadurch gekennzeichnet, daß an ihre (höchstens zweistellige) Abkürzung ein „$" angehängt wird. Es ist zulässig, in einem Programm gleichzeitig die numerische Variable „A" und die Stringvariable „A$" zu benutzen, unser Basic kann sie auseinanderhalten.

1)	ABS(	15)	DIM	29)	LEN(	43)	OUT	57)	SIN(
2)	AND	16)	DRAWTO	30)	LET	44)	PAGE	58)	SPC(
3)	ASC(	17)	END	31)	LIST	45)	PEEK(	59)	SQR(
4)	ATN(	18)	EXP(	32)	LLIST	46)	POKE	60)	STR$(
5)	CALL	19)	FRE(	33)	LOG(	47)	POS(	61)	STOP
6)	CHR$(	20)	FOR	34)	LPRINT	48)	PRINT	62)	TAB(
7)	CLEAR	21)	GOSUB	35)	MID$(	49)	READ	63)	TAN(
8)	CLRS	22)	GOTO	36)	MOVETO	50)	REM	64)	USR(
9)	CONT	23)	HEX(	37)	NEW	51)	RESTORE	65)	VAL(
10)	COS(	24)	IF	38)	NEXT	52)	RETURN	66)	WAIT
11)	CSAVE	25)	INP(	39)	NOT	53)	RIGHT$(	67)	? PRINT
12)	CLOAD	26)	INPUT	40)	NULL	54)	RND(		
13)	DATA	27)	INT(	41)	ON	55)	RUN		
14)	DEF FN	28)	LEFT$(	42)	OR	56)	SGN(		

Die Zahlen

Die interne Zahlendarstellung erfolgt mit sechs Stellen, plus Vorzeichen, plus zweistelligem, vorzeichenbehafteten Exponenten. Ausgegeben werden fünf gültige Stellen, die sechste wird gerundet. Daher ergibt die Eingabe von PRINT 123.456 <CR> die Anzeige „123.45" und die Eingabe von PRINT 123.456 + 123.456 <CR> die Anzeige „246.91". Allerdings erhält man bei der Eingabe von PRINT 1 ↑ 100000 <CR> dann auch wirklich den Wert 1.

Vor Zahlen wird ein positives Vorzeichen nicht dargestellt, sondern an dessen Stelle ein Leerzeichen ausgegeben. Es können Zahlen im Bereich von ca. –5E–39 bis ca. +5E+38 verarbeitet werden; diese scheinbar willkürlichen Grenzbereiche resultieren aus der internen binären Darstellungsform der Zahlen (drei Bytes für Mantisse und Vorzeichen, ein viertes Byte für den Exponenten plus Vorzeichen). Bei Gleitkommazahlen werden die Nachkommastellen nicht durch ein Komma, sondern durch einen Punkt abgetrennt. Das Komma ist in Basic dafür reserviert, zwei voneinander unabhängige Größen (z. B. Variablen in einer Aufzählung) gegeneinander abzugrenzen.

Bei Winkelfunktionen muß der Winkel in Bogenmaß (rad) angegeben werden. Die Umrechnung erfolgt durch Division des Winkelwertes (Grad) mit 180 und Multiplikation mit der Zahl PI (3.1415).

Operatoren

Operatoren nennt man alle die Zeichen, die veranlassen, daß eine Rechenart oder ein Vergleich durchgeführt wird. Operatoren möchten Zahlen oder Strings als Eingabe haben, operieren damit und werfen das Ergebnis als Zahl oder String aus. Einige Operatoren weichen in ihrer Schreibweise von der gewohnten algebraischen Notation ab, und durch den Einsatz von Klammern werden bestimmte Ausdrücke (z. B. im Nenner oder unter einer Wurzel stehende) zusammengefaßt.

+ Addition von Variablen; Strings können durch Addition aneinandergereiht werden

– Subtraktion von (numerischen) Variablen

* Multiplikation von (numerischen) Variablen

/ Division von (numerischen) Variablen

= hat außer der arithmetischen Bedeutung (Gleichheitszeichen) in Basic noch die Zuweisungsfunktion, d. h. einer links vom Gleichheitszeichen stehenden Variablen wird die rechts stehende Zahl, der Inhalt der rechtsstehenden Variablen, der Wert einer Funktion oder die Zeichenkette zugewiesen

< Vergleichsoperator für „kleiner als"

<= Vergleichsoperator für „kleiner als oder gleich" (Reihenfolge der beiden Operatoren beliebig)

> Vergleichsoperator für „größer als"

>= Vergleichsoperator für „größer als oder gleich" (Reihenfolge der beiden Operatoren beliebig)

<> Operator für „ungleich"

E definiert die nachgestellte Zahl (keine Variable!) als Exponent zur Basis Zehn; kann nicht allein, sondern nur in Verbindung mit einer vorangestellten Zahl (nicht Variablen!) verwendet werden
BEI EINGABEN MUSS DAS EXPONENTEN-E GROSS GESCHRIEBEN WERDEN!

↑ definiert die nachgestellte Variable als Exponenten zur vorangestellten Basis

PI (Kreiszahl) ist nicht fest in Basic gespeichert und muß bei Bedarf definiert werden: PI = 3.1415

Die Befehle und Funktionen

ABS(X)
Bildet den Absolutwert von X.

PRINT ABS(–523) <CR> Anzeige: 523
PRINT ABS(–5.1234*10 ↑ 4) <CR> Anzeige: 51234

X AND Y
Bildet auf Maschinen-Ebene bitweise die UND-Verknüpfung aus dem binären Äquivalent von X und Y, d. h. im Ergebniswort steht nur an der Stelle eine 1, an der die korrespondierenden Bits in beiden Operanden X *und* Y eine 1 hatten.

PRINT 101 AND 95 <CR> Anzeige: 69

ASC ("X") oder ASC (A$)
Erzeugt den zum ASCII-Zeichen "X" gehörenden Binärcode und gibt diesen dezimal aus.

PRINT ASC("H") <CR> Anzeige: 72
PRINT ASC("Meier") <CR> Anzeige: 77

ATN (X)
Bildet den Arkustangens vom Argument X. Das Ergebnis wird im Bogenmaß angegeben.

PRINT ATN(2) <CR> Anzeige: 1.1071

CALL X
Springt in das bei der Adresse X beginnende Unterprogramm in Maschinensprache, das mit dem Z80-Befehl „Return" (C9H) abgeschlossen sein muß. Die Adresse X ist dezimal zu verstehen (0...65535, keine Zeilennummer!); soll die Startadresse sedezimal genannt werden, ist dies mit HEX("X") möglich. Nach dem Rücksprung aus dem Maschinen-Unterprogramm geht die Programmausführung in Basic bei dem Befehl weiter, der hinter der CALL-Anweisung steht.

CHR$ (X)
Das der Zahl X entsprechende Zeichen nach ASCII wird ausgegeben. Bei Angaben für X, die außerhalb 0...255 (8-Bit-Wortlänge) liegen, wird mit der Fehlermeldung „?FC Fehler" (FUNCTION CALL ERROR) die Bereichsüberschreitung gemeldet.

PRINT CHR$(65) <CR> Anzeige: A

CLEAR
Setzt alle Variablen auf Null. Vorhandene Speicherplatzreservierungen werden nicht gelöscht.

CLEAR X
Setzt alle Variablen auf Null und reserviert X Bytes für die Variablen-Speicherung; X kann eine (positive) Variable sein, darf aber die Anzahl der freien Speicherstellen nicht überschreiten, sonst erfolgt die Fehlermeldung „?OM Fehler“ (OUT OF MEMORY). Zu kleine Reservierung ergibt „?OS Fehler“ (OUT OF STRING).

CLEAR 200 <CR>
PRINT FRE(A$) <CR> Anzeige: 200

CLRS
Löschen des Bildschirms

CONT
Veranlaßt den Interpreter, ein zuvor per STOP-Befehl abgebrochenes Programm beim nächstfolgenden Befehl fortzusetzen. In der Zwischenzeit dürfen die Variablen oder Zwischenergebnisse inspiziert werden, allerdings sind vor dem CONT-Befehl keine Änderungen im Programm zulässig, sonst erfolgt Fehlermeldung „?CN Fehler“ (CONTINUE ERROR). Beim INPUT-Befehl bewirkt die <CR>-Taste (Return-Taste) einen Abbruch (Meldung „o. k.“). Mit CONT wird der INPUT-Befehl wiederholt.

COS (X)
Bildet den Kosinus vom Argument X, das im Bogenmaß angegeben oder umgerechnet werden muß.

PRINT COS(60*3.1415/180) <CR>
Anzeige: .5002
PRINT COS(60) <CR> Anzeige: –.95241

CSAVE
Ausgabe des gesamten gespeicherten Programms auf Magnetband-Kassette (ohne Angabe eines Programmnamens und ohne Steuerung des Laufwerks). Es empfiehlt sich, das Programm doppelt abzuspeichern.

CLOAD
Einlesen eines auf Magnetband-Kassette gespeicherten Programms (ohne Angabe eines Programm-Namens und ohne Steuerung des Laufwerks). Ein zuvor im Speicher vorhandenes Programm wird gelöscht.

DATA X,Y,Z...
Definiert eine Datenzeile mit verschiedenen, durch Komma getrennten Zahlenwerten.

DATA A$, B$, C$...
Definiert eine Datenzeile mit verschiedenen, durch Komma getrennten Texten.

DEF FN A(X)=Y
Definiert den Variablennamen A als eine neue Funktion A, die die Rechenvorschrift Y (mathematischer Ausdruck) zusammenfaßt und diese beim Aufruf auf Variable anwendet. (X) ist ein Dummy-Argument, für das jeder beliebige alphanumerische Wert eingesetzt werden kann.

Beispiel: 10 DEF FN Q1(X) = X*B+B
20 INPUT A,B:PRINT FN Q1(A)

DIM X(z)
Reserviert für die numerische Variable "X" z Feldelemente, beginnend bei 0 und endend bei (z–1).

DIM X(i,j,k)
Reserviert für die numerische Variable "X" eine mehrdimensionale Matrix mit i*j*k Feldelementen, die jeweils mit dem Index 0 beginnen und mit (i–1) bzw. (j–1) bzw. (k–1) enden.

DIM A$(X)
Reserviert für die String-Variable "A$" X Feldelemente, beginnend bei 0 und endend bei (X–1).

DIM A$(i,j,k)
Reserviert für die String-Variable "A$" eine mehrdimensionale Matrix mit i*j*k Feldelementen, die jeweils mit dem Index 0 beginnen und enden bei (i–1) bzw. (j–1) bzw. (k–1).

DRAWTO X,Y
Zeichnet eine Gerade vom augenblicklichen Standpunkt des Cursors nach (x,y). Der Punkt (x,y) wird neuer Standpunkt. Der Bildschirm hat 512*256 Punkte. Die linke untere Bildecke besitzt die Koordinaten 0,0. Mit dem PAGE-Befehl wird die Schreibseite voreingestellt.

END
Zeigt dem Interpreter an, daß dieser die Programmausführung beenden und in die Basic-Anweisungs-Ebene zurückspringen soll. Dieses Statement ist entbehrlich, wenn am Programmende keine weiteren Programmzeilen folgen (z. B. die von Unterprogrammen); DATA-Anweisungen können hinter einem Programm stehen, ohne daß davor ein END eingefügt werden muß.

EXP (X)
Bildet die X-te Potenz zur Basis e (= 2.7182); bei zu großem X erfolgt die Fehlermeldung "?OV Fehler“ (OVERFLOW).

PRINT EXP(20) <CR>
Anzeige: 4.8516E+08

FRE (X)
Ermittelt ab Adresse 8800H die (dezimale) Anzahl freier Speicherplätze. Bei der SBC-2-Baugruppe mit zwei Speicherbausteinen (RAM) ergibt sich jeweils nach dem Kaltstart folgende Zahl:

PRINT FRE(0) <CR> Anzeige: 1784

FOR X = A TO Z STEP N
Definiert den Anfang einer Programmschleife, in der die Laufvariable X die Werte von A bis Z annehmen und bei jedem Durchlauf um die Schrittweite N erhöht werden soll; im Falle A = Z wird die Schleife einmal durchlaufen, und bei fehlender Angabe der Schrittweite N wird die Zahl 1 angenommen. Schleifen können geschachtelt werden.

GOSUB X
Spring in das bei Zeile X beginnende Unterprogramm, aus dem bei Erreichen des RETURN-Befehls (s. u.) automatisch der Rücksprung ins aufrufende Programm erfolgt; die Programmausführung geht mit der nächsten Basic-Zeile weiter, es muß in der Zeile GOSUB X der letzte Befehl in einer Zeile sein.

GOTO X
Setzt die Programmausführung bei Zeile X fort (unbedingter Sprungbefehl).

HEX ("X") oder **HEX (A$)**
Setzt die Sedezimalzahl ("Hexzahl") "X" in das dezimale Äquivalent um; mehr als vierstellige Angaben führen zu Fehlinterpretationen. Die Sedezimalzahl 0 – 7FFF ergibt 0...32767, 8000 – FFFF ergibt –32768...–1. Die sedezimalen Zeichen A...F müssen in Großbuchstaben eingegeben werden.

IF X=Y THEN (GOTO) Z
Setzt die Programmausführung bei Zeile Z fort, wenn die Bedingung X = Y erfüllt ist; andernfalls geht es bei der nächstfolgenden Basic-Zeile weiter (bedingter Sprung).

IF X<>Y THEN (GOTO) Z
Setzt die Programmausführung bei Zeile Z fort, wenn die Bedingung X<>Y (X ungleich Y) erfüllt ist; andernfalls geht es bei der nächstfolgenden Basic-Zeile weiter (bedingter Sprung).

IF X<Y THEN (GOTO) Z
Setzt die Programmausführung bei Zeile Z fort, wenn die Bedingung X<Y erfüllt ist, andernfalls geht es bei der nächstfolgenden Basic-Zeile weiter (bedingter Sprung).

INP (X)
Liest Daten von demjenigen Eingabe-Kanal ein, dem die dezimale Adresse X (0...255) zugeordnet ist. Die Angabe einer sedezimalen Adresse ist durch HEX ("X") möglich.

INPUT X
Dieser Befehl ist nicht im Direkt-Modus (ohne Zeilennummer) anwendbar. Die CR-Taste (Return-Taste) bricht die Programmausführung ab (Fortsetzung durch Eingabe von CONT). Der Befehl erwartet die Eingabe einer numerischen Variablen, die anschließend unter der Bezeichnung "X" geführt wird. Es ist darauf zu achten, daß Nachkomma-Stellen nicht durch ein Komma, sondern einen Punkt abgetrennt werden, weil das Komma zur Trennung zweier aufeinanderfolgender Eingaben dient (Fehlermeldung „zu viel“). Nicht-numerische Eingaben (z. B. Buchstaben oder Sonderzeichen) werden zurückgewiesen (Fehlermeldung „neue Eingabe“).

INPUT X,Y
Erwartet die Eingabe zweier numerischer Variablen, die durch ein Komma voneinander getrennt werden müssen und anschließend unter der Bezeichnung "X" "Y" geführt werden. Die Reaktionen auf Falscheingaben erfolgen sinngemäß wie bei INPUT X.

INPUT "ABC"; X
Wie INPUT X, aber mit vorheriger Ausgabe des Textes "ABC" auf dem Bildschirm, gefolgt vom Fragezeichen. Dies ist eine recht elegante Eingabeform, weil der Computer regelrecht nach der zu einem Text (z. B. „Lastwiderstand=?“) gehörenden Zahl „fragt“.

INPUT A$
Erwartet die Eingabe einer Zeichenkette mit max. 79 Zeichen, die beliebige Zeichen enthalten darf, also auch Ziffern, jedoch kein Komma, weil das zur Trennung zweier aufeinanderfolgender Eingaben dient. Soll der String auch ein Komma enthalten, ist er in Anführungszeichen zu setzen.

INT (X)
Liefert bei einer Gleitkommazahl X den nächstkleineren, ganzzahligen Zahlenwert (Integer).

PRINT INT(123.55) Anzeige: 123
PRINT INT(–0.5) Anzeige: –1

LEFT$ (A$,n)
Spaltet vom String A$ die ersten n Zeichen ab. Wenn der String nicht n Zeichen lang ist, werden entsprechend weniger genommen, ohne daß eine Fehlermeldung erfolgt. Zahlen für n<=0 ergeben die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR).

PRINT LEFT$("MEIER",3) <CR>
Anzeige: MEI

LEN (A$)
ermittelt die Anzahl der im String A$ enthaltenen Zeichen.

PRINT LEN("MEIER") <CR> Anzeige: 5

LET X=ABC
Weist der numerischen Variablen X den rechts vom Gleichheitszeichen stehenden Wert zu. Der Begriff „LET“ ist hierbei zwar entbehrlich, kann unter Umständen aber die Übersichtlichkeit eines Programms erhöhen.

LET A$ = "ABC"
Weist der String-Variablen A$ die in Anführungszeichen stehende Zeichenkette zu. Auch hier ist der Begriff "LET" entbehrlich, dient aber u. U. der Übersichtlichkeit eines Programmes.

LIST
Bewirkt die Ausgabe des gesamten gespeicherten Programms zeilenweise auf dem Bildschirm und kann mit der Taste <ESC> abgebrochen werden.

LIST X
Wie LIST, jedoch beginnend bei der Zeilennummer X. Um nur einen kleinen Programmausschnitt ausgeben zu lassen, gibt man "LIST X" und <CR> ein, hält die CTRL-Taste während der Return-Auslösung gedrückt und betätigt sofort (zusätzlich zur CTRL-Taste) zum Anhalten die Taste "S" und zum weiteren Anschauen die Taste "Q".

LLIST
Wie LIST, nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

LLIST X
Wie LIST X, nur erfolgt die Ausgabe auf dem Drucker.

LOG (X)
Bildet den natürlichen Logarithmus von X (zur Basis e = 2.7182); bei negativem X erfolgt die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR); bei zu großem X die Meldung "?OV Fehler" (OVERFLOW).

PRINT LOG(1000) <CR>
Anzeige: 6.9077

LPRINT X
Wirkt genauso wie PRINT, nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.
Für LPRINT gilt die abkürzende Form des Fragezeichens nicht.
Das diesem Befehl vorangestellte "L" steht als Abkürzung für engl. „Lineprinter“ (Zeilendrucker).

LPRINT "XYZ"
Wie PRINT "XYZ", nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

LPRINT A$
Wie PRINT A$, nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

LPRINT X;Y
Im Gegensatz zu PRINT X;Y erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

LPRINT X,Y
Im Gegensatz zu PRINT X,Y erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

MID$ (A$,n,m)
Löst aus dem String A$, beginnend beim n-ten Zeichen, m Zeichen heraus. Wenn nicht n oder m Zeichen vorhanden sind, werden entsprechend weniger genommen, ohne daß eine Fehlermeldung erfolgt. Zahlen für n bzw. m <= 0 ergeben die Meldung "?FC Fehler" (FUNCTION CALL ERROR).

PRINT MID$("ABCDEFGH,2,3) <CR>
Anzeige: BCD

MOVETO X,Y
Nur bei Grafikausgabe wird der Bildpunkt mit den Koordinaten X,Y neuer Cursor-Standpunkt, von dem aus mit dem Befehl DRAWTO U,V eine Linie zum Bildpunkt an der Stelle U,V gezeichnet wird.

mikrocomputer schritt für schritt

NEW
Initialisiert den Interpreter neu (Buffer und Variablen löschen, internen Stack definieren u. a.) und wirkt wie das Löschen des Programmspeichers. Tatsächlich aber wird der Programmspeicher nicht gelöscht, sondern an den Beginn des Programmspeichers (die Startadresse steht in den RAM-Zellen 8857H/58H) werden zwei Bytes "00" eingeschrieben. Außerdem wird das Ende des Programmspeichers (die Endadresse steht in den RAM-Zellen 8887H/88H) hinter das zweite gelöschte Byte gesetzt (also um zwei Plätze höher als der Programmanfang). Das bedeutet, daß das Programm (bis auf die ersten beiden Bytes) noch im Speicher steht, vom Interpreter aber nicht mehr ausgeführt werden kann, weil Anfangs- und Endmarkierung durch die NEW-Anweisung zusammengelegt worden sind. Beim Einbau in ein Programm wirkt dieser Befehl wie ein Software-Selbstmord: Das Programm löscht sich quasi selbst aus.

NEXT X
Definiert das Ende der mit FOR...TO... STEP begonnenen Programmschleife. Die Nennung der Laufvariablen X kann hierbei entfallen.

NOT X
Invertiert im binären Äquivalent von X jedes einzelne Bit; durch die dezimale Anzeige der Ergebnisse kommt es oft zu Mißverständnissen, weil das Ergebnis von "NOT 55" gleich –56 ist. Das liegt daran, daß der Interpreter ein HIGH im Höchstwertigen Bit eines Ergebniswortes als negatives Vorzeichen wertet.

NULL X
Der Befehl NULL X <CR> verschiebt den linken Rand auf dem Bildschirm um X Zeichen nach rechts. Längere Zeichenketten werden auf der linken Bildschirmseite fortgesetzt.

ON X GOTO A,B,C...
Setzt die Programmausführung bei Zeile A fort, wenn X = 1 ist, bei Zeile B, wenn X = 2 ist, bei Zeile C, wenn X = 3 ist, usw. (bedingte Verzweigung).

X OR Y
Bildet auf Maschinen-Ebene bitweise die ODER-Verknüpfung aus dem binären Äquivalent von X und Y, d. h. im Ergebniswort steht überall dort eine 1, wo an der korrespondierenden Stelle der Operanden X oder Y eine 1 war.

PRINT 37 OR 16 <CR> Anzeige: 53

OUT X,Y
Gibt den zu Y gehörenden Zahlenwert (in binärer Form) an demjenigen Ausgabe-Kanal aus, dem die Adresse X zugeordnet ist; X ist hierbei dezimal zu verstehen (0...255; die Angabe einer sedezimalen Adresse ist durch HEX("X") bzw. HEX("Y") möglich.

PAGE N,M
Der GDP64 wird voreingestellt. Dabei wird auf die Seiten N geschrieben und die Seite M angezeigt. Voreingestellt ist die Seite 1 (für Textmodus). Auf diese Seite wird geschaltet, wenn Basic sich mit "o. k." meldet. Nach diesem Befehl darf kein Input-Befehl folgen, da sonst die Seiteneinstellung nicht mehr stimmt. Mit dem Befehl PAGE 0,0 in einem Basic-Grafik-Programm wird der Bildaufbau sichtbar.

PEEK (X)
Liest den Inhalt der Speicherzelle mit der Adresse X; X ist hierbei dezimal zu verstehen (0...65535; die Angabe einer sedezimalen Adresse ist durch HEX ("X") möglich).

PRINT PEEK(HEX("87C5")) Anzeige: 10

POKE X,Y
Schreibt den Wert Y in die Speicherzelle mit der Adresse X; X und Y sind dabei dezimal zu verstehen (0...65535 bzw. 0...255; die Angabe sedezimaler Zahlen ist durch HEX ("X") bzw. HEX ("Y") möglich.

POKE HEX(„87C5"),2 <CR>
Cursorblinkfrequenz schneller

POS (X)
Ermittelt die Cursor-Position in der laufenden Zeile, nennt also die Anzahl der bereits ausgegebenen Zeichen. (X) ist ein Dummy-Argument, für das jeder beliebige alphanumerische Wert eingesetzt werden kann.

PRINT X
Dient im Direkt-Modus dazu, nach Return unmittelbar eine Ergebnisanzeige auf dem Bildschirm zu erzeugen (ohne vorherigen Programmstart), z. B. das Ergebnis einer Rechenoperation oder die Darstellung von Zwischenergebnissen bzw. Variablen nach einer Programmunterbrechung. Dabei kann "X" eine im Programm verwendete Variable sein oder eine Rechenvorschrift (z. B. 275*1.14) oder bei "X$" eine Zeichenkette oder eine beliebige Zeichenfolge, die dann in Anführungszeichen zu setzen ist. Beim Einsatz dieses Befehls in einem Programm ergibt sich eine Fülle von Varianten. Zur Abkürzung kann man anstelle von "PRINT" einfach ein Fragezeichen "?" eingeben.

PRINT "XYZ"
Bewirkt die Ausgabe der in Anführungszeichen stehenden Zeichenkette auf dem Bildschirm.

PRINT A$
Bewirkt die Ausgabe der zur String-Variablen A$ gehörenden Zeichenkette auf dem Bildschirm.

PRINT X;Y
Bewirkt die Ausgabe der entsprechenden Zahlenwerte für X und Y hintereinander. Somit können auch numerische und String-Variablen zusammen ausgegeben werden, z. B. das Ergebnis einer Rechnung mit einem passenden Text. Zwischen beide Ausgaben wird ein Leerzeichen zur Trennung eingefügt; vor positiven Zahlenwerten steht ein weiteres Leerzeichen, weil das positive Vorzeichen unterdrückt wird.

PRINT X,Y
Bewirkt die Ausgabe der entsprechenden Zahlenwerte in einer Zeile, wobei die zweite (und jede folgende) Ausgabe bei der nächsten Tabulator-Position anfängt (neue Tabulator-Position: alle 14 Spalten).

READ N
Ruft das jeweils nächste Daten-Element (in diesem Fall eine Zahl) ab; bei jeder Ausführung des READ-Befehls wird ein interner Daten-Pointer um Eins erhöht, um für den folgenden Zugriff das nächste Element zu adressieren. Findet der Interpreter kein Daten-Element mehr, weil mehr READ-Befehle ausgeführt wurden als Daten bereitgestellt sind, erfolgt die Fehlermeldung "?OD Fehler" (OUT OF DATA).

READ N$
Wie READ N, jedoch hier Abruf von Texten.

REM
Kleinbuchstaben bleiben nach dem REM-Befehl erhalten. Dieser definiert die laufende Zeile als Kommentarzeile, d. h. nach "REM" kann jeder beliebige, erläuternde Text stehen. Der Interpreter übergeht eine Kommentarzeile bei der Programmausführung, jedoch kann eine solche Zeile als Sprungziel dienen, sollte aber nicht, weil REM-Zeilen keine Befehle enthalten.

10 REM Gitternetz oder 10 PRINT A: REM Ausgabe

RESTORE
Bewirkt das Rücksetzen des DATA-Zählers, der mit jedem READ-Befehl um Eins erhöht wird und damit stets auf die nächste, per DATA definierte Konstante weist. Wird bei READ keine durch DATA definierte Variable mehr gefunden, erfolgt die Fehlermeldung "?OD Fehler" (OUT OF DATA); das vorherige Rücksetzen per RESTORE vermeidet dies.

RETURN
Schließt ein BASIC-Unterprogramm ab und bewirkt den Rücksprung an die zuvor mit dem Befehl GOSUB verlassene Stelle im aufrufenden Programm (s. o.).

RIGHT$ (A$,n)
Spaltet vom String A$ die letzten n Zeichen ab. Wenn der String nicht n Zeichen lang ist, werden entsprechend weniger genommen, ohne daß eine Fehlermeldung erfolgt. Zahlen für n<=0 ergeben die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR).

RND (X)
Erzeugt eine Gleitkomma-Pseudo-Zufallszahl zwischen 0...1; es ist sichergestellt, daß bei verschiedenen Programmdurchläufen nicht jedesmal dieselbe Zahl bzw. dieselbe Zahlenfolge auftritt. (X) ist ein Dummy-Argument. Negative Zahlen ergeben konstante Pseudozufallszahlen.

PRINT RND(–1) <CR>
Anzeige: 7.6594E-06

RUN
Veranlaßt den Interpreter, ein gespeichertes Programm auszuführen, beginnend bei der niedrigsten Zeilennummer.

RUN X
Veranlaßt den Interpreter, ein gespeichertes Programm auszuführen, beginnend bei der Zeilennummer X.

SGN (X)
Liefert das Vorzeichen von X; +1 bei positivem X, –1 bei negativem X und Null bei X = 0 (Signum-Funktion).

SIN (X)
Bildet den Sinus vom Argument X, das im Bogenmaß angegeben oder umgerechnet werden muß.

PRINT SIN(45*3.1415/180) <CR>
Anzeige: .70709
PRINT SIN(45) <CR> Anzeige: .8509

SPC (X)
Rückt den Cursor um X Stellen nach rechts und füllt den Zwischenraum mit Leerzeichen (Blanks) auf.

SQR (X)
Bildet die Quadratwurzel aus dem (positiven) Argument X. Bei negativem X erfolgt die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR) für die Bereichsüberschreitung und bei zu großem X wird "?OV Fehler" (OVERFLOW) angezeigt. Die Bezeichnung "SQR" steht als Abkürzung für engl. "Square Root" = Quadratwurzel.

PRINT SQR(2) <CR> Anzeige: 1.4142

STR$ (X)
Wandelt die numerische Variable X in eine String-Variable um und ermöglicht damit die Anwendung von String-Operationen auf Zahlen. Nach erfolgter Umwandlung kann mit der neu definierten String-Variablen keine mathematische Operation mehr durchgeführt werden, auch wenn es sich augenscheinlich um eine Zahl handelt.

STOP
Bewirkt nach der Ausführung dieses Befehls die Unterbrechung des Programms mit der Meldung "abgebrochen in Zeile xyz"; anschließend sind die Inspektion und Modifikation von Variablen mit darauffolgender Fortsetzung des Programms möglich (per CONT), allerdings darf dabei das Programm selbst nicht modifiziert werden (programmierte Programmunterbrechung).

TAB (X)
Rückt den Cursor vom linken Bildrand aus um X Stellen nach rechts. Im Beispiel kann daher der zweite TAB-Befehl nicht ausgeführt werden.

PRINT TAB(10);"*";TAB(5);"*" <CR>
Anzeige: **

TAN (X)
Bildet den Tangens vom im Bogenmaß angegebenen Wert X.

PRINT TAN(45*3.1415/180) <CR>
Anzeige: .99995
PRINT TAN(45) <CR> Anzeige: 1.6197

USR (X)
Ruft die bei der Adresse X beginnende Anwender-Funktion auf und übergibt (im Gegensatz zu CALL) das Ergebnis im Gleitkomma-Akkumulator des Interpreters (Adressen 886E...8870H für Mantisse und Vorzeichen und Adresse 8871H für den vorzeichenbehafteten Exponenten).

VAL (A$)
Wandelt die String-Variable A$ in eine numerische Variable um und ermöglicht anschließend wieder die Anwendung mathematischer Operationen. Von A$ wird allerdings nur derjenige numerische Anteil berücksichtigt (Folge von Zahlen), der keine ASCII-Zeichen enthält, d. h. alle Zeichen, die im ursprünglichen String A$ rechts vom ersten nicht-numerischen Zeichen stehen, werden bei der Typ-Umwandlung nicht berücksichtigt.

PRINT VAL("1234ABCD") <CR>
Anzeige: 1234
PRINT VAL("ABC12") <CR>
Anzeige: 0

WAIT X,Y,Z
Wie INP X, aber mit anschließender Exklusiv-ODER-Verknüpfung mit Z, gefolgt von der UND-Verknüpfung mit Y. Der Interpreter führt den nächsten Befehl erst dann aus, wenn das so entstandene Ergebnis ungleich Null ist. Im Falle Z = 0 (oder bei fehlender Z-Angabe) werden die eingelesenen Daten nur mit Y UND-verknüpft. Die angegebenen Werte für X, Y und Z sind dezimal zu verstehen (0...255); die Angabe von sedezimalen Operanden ist durch HEX ("X") bzw. HEX ("Y") bzw. HEX ("Z") möglich.
Der Befehl WAIT 70,4,0 bewirkt eine Warteschleife, die erst verlassen wird, wenn Bit 2 des Statusregisters mit der Adresse 70H des Grafikprozessors EF9366 den Wert 1 besitzt.

Fehlererkennung und Fehlermeldung

Bei der Umsetzung und Ausführung eines gespeicherten Programms prüft der Interpreter ständig, ob bei der Formulierung der einzelnen Befehle das vorgeschriebene Eingabeformat eingehalten worden ist (z. B. Setzen von Klammern, Anhängen eines Dollar-Zeichens o. ä.).

Derartige Syntax-Fehler sind für das Programm ohne Schwierigkeiten erkennbar, weil es die Eingaben nur mit einem vorgegebenen Muster zu vergleichen braucht.
Darüber hinaus gibt es Fehler des Bedieners, die eine Programmausführung unmöglich machen (z. B. wenn der Programm- oder Variablenspeicher voll ist und keine Daten mehr aufgenommen werden können). Ebenso gehört die Prüfung auf verbotene Rechengänge (Teilen durch Null, Wurzel aus negativer Zahl ziehen), fehlende Definitionen oder Bereichsüberschreitungen zu den Überwachungsaufgaben des Interpreters.
Fehlermeldung: ?FF FEHLER (IN ZEILE XYZ)

Es wurde einer der obengenannten Fehler erkannt, für dessen Identifikation ein zweistelliger Fehlercode "FF" ausgegeben wird.
Im Direkt-Modus erfolgt nur die Ausgabe "?FF Fehler", während im Programmlauf auch noch die Zeilennummer angegeben wird, in der der Fehler auftritt: "?FF Fehler in Zeile XYZ"; bis auf die (international üblichen) zweistelligen Fehlercodes erfolgt diese Meldung in deutsch. Nach einer derartigen Fehlermeldung wird ein Programmlauf sofort abgebrochen, und der Interpreter geht zurück in den BASIC-Anweisungs-Modus (o.k.-Meldung).

Zuviel

Wenn bei der INPUT-Anweisung, die eine Bediener-Eingabe verlangt, mehr Eingaben gemacht werden als angefordert (mehrere Eingaben werden durch Komma voneinander getrennt), dann erfolgt die Meldung „zuviel". In diesem Fall ignoriert der Interpreter die überzähligen Eingaben und fährt mit der Programmausführung fort.
Diese Meldung ergeht auch dann, wenn eine Dezimalzahl anstelle des Dezimalpunktes ein Komma enthält, da der Interpreter dies als *zwei* Eingaben versteht (Trennzeichen Komma).

Neue Eingabe

Wenn bei Eingaben der falsche Variablen-Typ gewählt wird (bei erwarteten numerischen Daten die Eingabe von Buchstaben oder umgekehrt), dann erfolgt die Meldung „neue Eingabe". In diesem Fall ignoriert der Interpreter die falschen Eingaben und erwartet die Eingabe des richtigen Datentyps; danach fährt er mit der Programmausführung fort.

Liste und Bedeutung der Fehlermeldungen

BS BAD SUBSCRIPT, Undefiniertes Feldelement: falsche Indizierung; ein aufgerufenes Matrix-Element liegt außerhalb der durch DIM festgelegten Grenzen.

CN CONTINUE ERROR, kein CONT möglich: Die Fortsetzung eines zuvor unterbrochenen Programms per CONT ist nicht möglich, weil entweder ein Fehler vorliegt oder das Programm selbst zwischenzeitlich modifiziert worden ist.

DD DOUBLE DIMENSION, Mehrfachdefinition eines Feldes: Dasselbe Feld wird im Programm noch einmal dimensioniert.

FC FUNCTION CALL ERROR, Rechenfehler: Bei einem Funktionsaufruf liegt ein Parameter außerhalb des zulässigen Bereichs, z. B. beim Wurzelziehen aus einer negativen Zahl.

ID ILLEGAL DIRECT, als Direktbefehl nicht erlaubt: Die gewählte Anweisung ist im Direkt-Modus nicht zulässig; eine Funktions-Definition beispielsweise kann nur innerhalb eines Programms, nicht aber im Direkt-Modus aufgerufen werden.

LS LONG STRING, String zu lang: Ein String überschreitet die maximal zulässige Länge von 255 Zeichen.

NF NEXT WITHOUT FOR, Next ohne for: Eine Programmschleife ist unvollständig programmiert worden.

OD OUT OF DATA, zu wenig Daten: Es wurden mehr READ-Befehle ausgeführt als Daten bereitgestellt waren. Entweder müssen mehr Daten eingeführt werden oder der Daten-Pointer ist mittels RESTORE an den Beginn der Datenreihe zurückzusetzen.

OM OUT OF MEMORY, Variablenspeicher voll: Ein Bereich des Arbeitsspeichers ist voll; das kann ein zu langes Programm sein, eine Überschreitung des Variablen-Speichers oder auch eine Überfüllung des vom Interpreter benutzten Stacks (z. B. durch zu viele ineinander verschachtelte Unterprogramme oder FOR...NEXT-Schleifen).

OS OUT OF STRING-SPACE, Stringspeicher voll: Der für Strings reservierte Speicherbereich ist voll; das kann durch zu viele oder zu lange Strings passieren.

OV OVERFLOW, Überlauf beim Rechnen: Das Ergebnis einer mathematischen Operation überschreitet den max. möglichen Zahlenbereich.

RG RETURN WITHOUT GOSUB, Return ohne gosub: Es taucht ein RETURN-Befehl auf, ohne daß zuvor ein Unterprogramm-Aufruf erfolgt ist.

SN SYNTAX ERROR, Syntax-Fehler: Es liegt eine Verletzung des vorgeschriebenen Eingabe-Formats vor, z. B. die Eingabe "CHR(X)" statt "CHR$(X)".

ST STRING TOO COMPLEX, Fehler bei Stringverarbeitung: Ein String ist zu lang; er muß kürzer gefaßt oder in mehrere kürzere aufgeteilt werden.

TM TYPE MISMATCH, unterschiedliche Variablentypen: In einer Zuweisung kollidieren unterschiedliche Variablen-Typen; anstelle einer erwarteten numerischen Variablen wurde ein String übergeben oder umgekehrt.

UF UNDEFINED FUNCTION, undefinierte Funktion: Für eine im Programm aufgerufene Funktion fehlt zuvor die entsprechende Definition.

US UNDEFINED STATEMENT, Sprungziel fehlt: Es wurde eine falsche, nicht im Basic-Befehlssatz enthaltene Anweisung eingegeben (bzw. eine richtig gemeinte Anweisung wurde falsch geschrieben) oder es wurde im Programm ein Sprungziel aufgerufen, das gar nicht existiert.

/0 DIVISION BY ZERO, Teilung durch 0: Jemand hat verbotenerweise versucht, durch Null zu teilen.

Kleine Beispielprogramme zur Grafik

Die nächsten beiden Seiten zeigen, wie man von Basic aus die GDP64 ansprechen kann.


```
  1 REM GITTERNETZ 25*25 FELDER
  5 CLRS
 10 PAGE 0,0                          :REM Schreibseite 0
 20 FOR I = 0 TO 500 STEP 20          :REM senkrechte Linien
 30 MOVETO I,0
 40 DRAWTO I,250
 50 NEXT
 60 FOR I = 0 TO 500 STEP 10          :REM waagrechte Linien
 70 MOVETO 0,I
 80 DRAWTO 500,I
 90 NEXT
100 GOTO 100
```

```
2. Zufallsquadrate  (Bild 3)

  5 CLRS:PAGE 0,0
 10 X = RND(1)*511
 20 Y = RND(1)*250
 30 MOVETO X,Y
 40 GOSUB 100
 50 GOTO 10
100 REM WUERFEL
110 X = X + 10:GOSUB 200
120 Y = Y +  5:GOSUB 200
130 X = X - 10:GOSUB 200
140 Y = Y -  5:GOSUB 200
200 DRAWTO X,Y:RETURN
```

```
3. Sinusschwingungen  (Bild 4)

10 CLRS:PI = 3.1415
20 INPUT"Anzahl der Schwingungen";S: S = S*2
30 PAGE 0,0:FOR X = 0 TO 511
40 Y = INT(SIN(X/511*PI*S)*120+120)
50 MOVETO X,120:DRAWTO X,Y
60 NEXT
70 POKE HEX("87C5"),0                 :REM Cursor aus
```

```
4. Sinusgraphik  (Bild 5)

10 CLRS: PAGE 0,0
20 FOR X = 0 TO 511
25 PI = 3.1415
30 Y = INT(SIN(X/511*PI*4)*COS(X/511*17*PI)*100+100)
40 Y1 = INT(SIN(X/511*PI*4)*100+100)
50 MOVETO X,100                       :REM Startpunkt
60 DRAWTO X,Y                         :REM Zielpunkt
70 MOVETO X,Y1:DRAWTO X,Y1
75 NEXT X
80 POKE HEX("87C5"),0
```

Bild 1. Einige Beispiele, wie man in Basic mit der Grafik umgeht

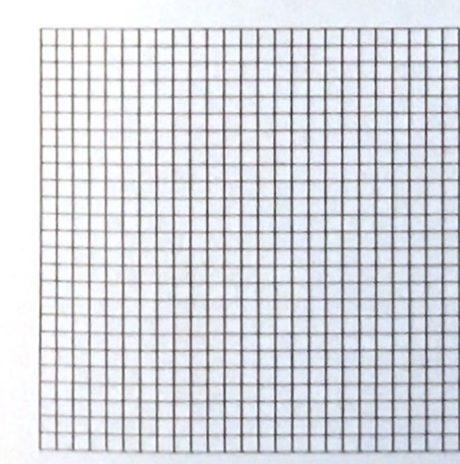


Bild 2. Das Gitternetz



Bild 3. Das Ergebnis eines Laufes von Zufallsquadraten

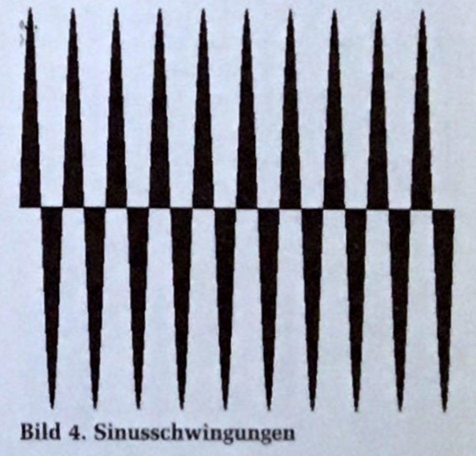


Bild 4. Sinusschwingungen

```
10 CLEAR 100: CLRS
20 GDP=HEX("70")
30 INPUT "TEXT:";A$
40 INPUT "koordinaten X,Y:";X,Y
50 PAGE 0,0
60 OUT GDP+3,7+7*16:               REM Port 73 = Vergrößerung
70 MOVETO X,Y
80 FOR I = 1 TO LEN(A$)
85 OUT GDP,ASC(MID$(A$,I,1)):      REM Ausgabe der ASCII-Werte PORT 70
90 WAIT GDP,4,0:                   REM Warteschleife bis GDP fertig
100 NEXT I
110 OUT GDP+3,1+16                 REM alte Schriftgröße
120 POKE HEX("87C5"),0:            REM Cursor aus
130 INPUT"Nochmal";X$
140 IF X$ = "J" THEN 10


6. Hex-Monitor  (Bild 7)

10 REM Hex-Monitor
20 CLEAR 100:DIM H$(16):DATA 0,1,2,3,4.5,6,7,8,9,A,B,C,D,E,F
30 FOR I = 0 TO 15:READ H$(I):NEXT I
50 TZ$="0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F"
60 TS$="dez.  hex.   ":CLRS
100 PRINT"Speicher anschauen   =  1":PRINT
110 PRINT"Speicher aendern     =  2":PRINT
130 INPUT"Zahl";A: IF A>2 OR A<1 THEN 60
1000 CLRS: INPUT"ab Hex-Adr";B$:B = HEX(B$):PRINT
1015 IF B<0 THEN B=65536+B
1016 IF A=2 THEN 2000
1018 B=INT(B/16)*16
1020 INPUT"bis:";C$:C = HEX(C$):CLRS
1025 IF C<0 THEN C = 65536 + C
1030 PRINT TS$;TZ$
1040 FOR I = B TO C
1080 IF I/16 = INT(I/16) THEN PRINT:ZR=I:GOSUB 6010
1085 D=PEEK(I):S=0:GOSUB 5017
1090 NEXT I:PRINT:INPUT"nochmal";X$
1100 IF LEFT$(X$,1) ="j" THEN 1000
1110 GOTO 60
1500 REM AENDERN
2000 ZR=B:GOSUB 6010
2003 D=PEEK(B):S=0:GOSUB 5017
2005 INPUT"hex";F$:F=HEX(F$):IF F>255 THEN 2005
2010 POKEB,F:B=B+1:GOTO 2000
3000 REM DEZ-HEX
5000 S = INT(D/256):gosub 5100
5015 PRINT H$;
5017 S = D - S*256:GOSUB 5100
5020 PRINT H$ + "  ";:RETURN
5100 L = S AND 15: H = (S AND 240)/16
5150 H$ = H$(H)+H$(L):RETURN
6000 REM Format
6010 ZR$=STR$(ZR ):PRINT MID$(ZR$,2,6);SPC(8-LEN(ZR$));
6020 D=ZR:GOSUB 5000
6030 RETURN
```

Bild 1b.

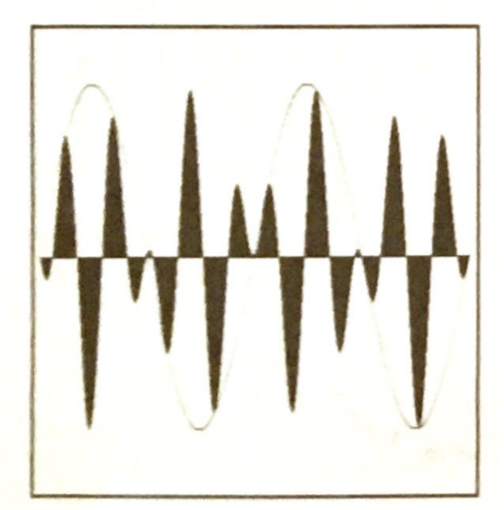


Bild 5. Sinusgrafik



Bild 6. Texte können vergrößert werden

Bild 7. Die Ausgabe des Hexmonitors

Kurze Maschinenprogramme lassen sich mit unserem Monitor schnell und einfach mit den hexadezimalen Operationscodes schreiben. Heinz-Peter Curdts hat sich die Mühe gemacht, die Z80-Maschinenbefehle übersichtlich zusammenzustellen

8-bit-Ladebefehle

	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(nn)	n
LD A,..	7F	78	79	7A	7B	7C	7D	7E	0A	1A	3AXXXX	3EXX
LD B,..	47	40	41	42	43	44	45	46				06XX
LD C,..	4F	48	49	4A	4B	4C	4D	4E				0EXX
LD D,..	57	50	51	52	53	54	55	56				16XX
LD E,..	5F	58	59	5A	5B	5C	5D	5E				1EXX
LD H,..	67	60	61	62	63	64	65	66				26XX
LD L,..	6F	68	69	6A	6B	6C	6D	6E				2EXX
LD (HL),..	77	70	71	72	73	74	75					36XX
LD (BC),..	02											
LD (DE),..	12											
LD (nn),..	32XXXX											

	A	B	C	D	E	H	L
LD ..,(IX+d)	DD7EXX	DD46XX	DD4EXX	DD56XX	DD5EXX	DD66XX	DD6EXX
LD ..,(IY+d)	FD7EXX	FD46XX	FD4EXX	FD56XX	FD5EXX	FD66XX	FD6EXX
LD (IX+d),..	DD77XX	DD70XX	DD71XX	DD72XX	DD73XX	DD74XX	DD75XX
LD (IY+d),..	FD77XX	FD70XX	FD71XX	FD72XX	FD73XX	FD74XX	FD75XX

LD (IX+d),n DD36XXXX LD (IY+d),n FD36XXXX

		S	Z	H	P/V	N	C
LD A,I	ED57	*	*	0	*	0	-
LD A,R	ED5F	*	*	0	*	0	-
LD I,A	ED47	-	-	-	-	-	-
LD R,A	ED4F	-	-	-	-	-	-

16-bit-Ladebefehle

	BC	DE	HL	SP	IX	IY
LD ...,nn	01XXXX	11XXXX	21XXXX	31XXXX	DD21XXXX	FD21XXXX
LD ...,(nn)	ED4BXXXX	ED5BXXXX	2AXXXX	ED7BXXXX	DD2AXXXX	FD2AXXXX
LD (nn),...	ED43XXXX	ED53XXXX	22XXXX	ED73XXXX	DD22XXXX	FD22XXXX
LD SP			F9		DDF9	FDF9

	BC	DE	HL	AF	IX	IY
PUSH ..	C5	D5	E5	F5	DDE5	FDE5
POP ..	C1	D1	E1	F1	DDE1	FDE1

EX (SP),HL	E3	EX DE,HL	EB	
EX (SP),IX	DDE3	EX AF,AF'	08	
EX (SP),IY	FDE3	EXX	D9	BC-BC' DE-DE' HL-HL'

Blocktransfer- und Suchbefehle

		S	Z	H	P/V	N	C	
LDI	EDA0	-	-	0	*	0	-	LD(DE),(HL);INC HL;INC DE;DEC BC
LDIR	EDB0	-	-	0	0	0	-	wie LDI,wiederholen bis BC=0
LDD	EDA8	-	-	0	*	0	-	LD(DE),(HL);DEC HL;DEC DE;DEC BC
LDDR	EDB8	-	-	0	0	0	-	wie LDD,wiederholen bis BC=0
CPI	EDA1	*	*	*	*	1	-	LD A,(HL);INC HL;DEC BC
CPIR	EDB1	*	*	*	*	1	-	wie CPI,wiederholen bis BC=0 oder Zeichen gefunden
CPD	EDA9	*	*	*	*	1	-	LD A,(HL);DEC HL;DEC BC
CPDR	EDB9	*	*	*	*	1	-	wie CPD,wiederholen bis BC=0 oder Zeichen gefunden

Sprungbefehle

	Z	NZ	C	NC	PE	PO	M	P
JP	CAXXXX	C2XXXX	DAXXXX	D2XXXX	EAXXXX	E2XXXX	FAXXXX	F2XXXX
CALL	CCXXXX	C4XXXX	DCXXXX	D4XXXX	ECXXXX	E4XXXX	FCXXXX	F4XXXX
RET	C8	C0	D8	D0	E8	E0	F8	F0
JR	28XX	20XX	38XX	30XX				

	unbedingt	(HL)	(IX)	(IY)
JP	C3XXXX	E9	DDE9	FDE9
CALL	CDXXXX			
RET	C9			
JR	18XX			

RST	00	08	10	18	20	28	30	38
	C7	CF	D7	DF	E7	EF	F7	FF

DJNZ	10XX	DEC B;JR NZ,d
RETI	ED4D	zurueck vom Interrupt
RETN	ED45	zurueck vom nicht maskierbaren interrupt

CPU-Steuerbefehle

		S	Z	H	P/V	N	C	
NOP	00	-	-	-	-	-	-	Leerbefehl
HALT	76	-	-	-	-	-	-	
CCF	3F	-	-	*	-	0	*	Komplementiere Carry-Flag
SCF	37	-	-	0	-	0	1	Setze Carry-Flag
EI	FB	-	-	-	-	-	-	Interrupts freigeben
DI	F3	-	-	-	-	-	-	Interrupts sperren
IM 0	ED46	-	-	-	-	-	-	Interrupt-Modus 0
IM 1	ED56	-	-	-	-	-	-	Interrupt-Modus 1
IM 2	ED5E	-	-	-	-	-	-	Interrupt-Modus 2

Ein-/Ausgabebefehle

	A	B	C	D	E	H	L	S	Z	H	P/V	N	C
IN ..,(C)	ED78	ED40	ED48	ED50	ED58	ED60	ED68	*	*	0	*	0	-
OUT (C),..	ED79	ED41	ED49	ED51	ED59	ED61	ED69	-	-	-	-	-	-

		S	Z	H	P/V	N	C	
IN A,n	DBXX	-	-	-	-	-	-	
OUT n,A	D3XX	-	-	-	-	-	-	
INI	EDA2	?	*	?	?	1	-	IN (HL),(C);INC HL;DEC B
INIR	EDB2	?	1	?	?	1	-	wie INI,wiederholen solange B<>0
IND	EDAA	?	*	?	?	1	-	IN (HL),(C);DEC HL;DEC B
INDR	EDBA	?	1	?	?	1	-	wie IND,wiederholen solange B<>0
OUTI	EDA3	?	*	?	?	1	-	OUT (C),(HL);INC HL;DEC B
OTIR	EDB3	?	1	?	?	1	-	wie OUTI,wiederholen solange B<>0
OUTD	EDAB	?	*	?	?	1	-	OUT (C),(HL);DEC HL;DEC B
OTDR	EDBB	?	1	?	?	1	-	wie OUTD,wiederholen solange B<>0

8-bit Arithmetische und Logische Befehle

	B	C	D	E	H	L	(HL)	A	n	(IX+d)	(IY+d)	S	Z	H	P/V	N	C
ADD .	80	81	82	83	84	85	86	87	C6XX	DD86XX	FD86XX	*	*	*	*	0	*
ADC .	88	89	8A	8B	8C	8D	8E	8F	CEXX	DD8EXX	FD8EXX	*	*	*	*	0	*
SUB .	90	91	92	93	94	95	96	97	D6XX	DD96XX	FD96XX	*	*	*	*	1	*
SBC .	98	99	9A	9B	9C	9D	9E	9F	DEXX	DD9EXX	FD9EXX	*	*	*	*	1	*
AND .	A0	A1	A2	A3	A4	A5	A6	A7	E6XX	DDA6XX	FDA6XX	*	*	1	*	0	0
XOR .	A8	A9	AA	AB	AC	AD	AE	AF	EEXX	DDAEXX	FDAEXX	*	*	1	*	0	0
OR .	B0	B1	B2	B3	B4	B5	B6	B7	F6XX	DDB6XX	FDB6XX	*	*	1	*	0	0
CP .	B8	B9	BA	BB	BC	BD	BE	BF	FEXX	DDBEXX	FDBEXX	*	*	*	*	1	*
INC .	04	0C	14	1C	24	2C	34	3C		DD34XX	FD34XX	*	*	*	*	0	-
DEC .	05	0D	15	1D	25	2D	35	3D		DD35XX	FD35XX	*	*	*	*	1	-

		S	Z	H	P/V	N	C	
DAA	27	*	*	*	*	-	*	BCD-Korrektur im Akku
CPL	2F	-	-	1	-	1	-	Komplementiere Akku (1er-Komplement)
NEG	ED44	*	*	*	*	1	*	Komplementiere Akku (2er-Komplement)

16-bit Arithmetische und Logische Befehle

	BC	DE	HL	SP	IX	IY	S	Z	H	P/V	N	C
INC .	03	13	23	33	DD23	FD23	-	-	-	-	-	-
DEC .	0B	1B	2B	3B	DD2B	FD2B	-	-	-	-	-	-
ADD HL,.	09	19	29	39			-	-	*	-	0	*
ADC HL,.	ED4A	ED5A	ED6A	ED7A			*	*	*	*	0	*
SBC HL,.	ED42	ED52	ED62	ED72			*	*	*	*	1	*
ADD IX,.	DD09	DD19		DD39	DD29		-	-	*	-	0	*
ADD IY,.	FD09	FD19		FD39		FD29	-	-	*	-	0	*

Rotations- und Schiebebefehle

	B	C	D	E	H	L	(HL)	A	(IX+d)	(IY+d)
RR	CB18	CB19	CB1A	CB1B	CB1C	CB1D	CB1E	CB1F	DDCBXX1E	FDCBXX1E
RL	CB10	CB11	CB12	CB13	CB14	CB15	CB16	CB17	DDCBXX16	FDCBXX16
RRC	CB08	CB09	CB0A	CB0B	CB0C	CB0D	CB0E	CB0F	DDCBXX0E	FDCBXX0E
RLC	CB00	CB01	CB02	CB03	CB04	CB05	CB06	CB07	DDCBXX06	FDCBXX06
SRA	CB28	CB29	CB2A	CB2B	CB2C	CB2D	CB2E	CB2F	DDCBXX2E	FDCBXX2E
SLA	CB20	CB21	CB22	CB23	CB24	CB25	CB26	CB27	DDCBXX26	FDCBXX26
SRL	CB38	CB39	CB3A	CB3B	CB3C	CB3D	CB3E	CB3F	DDCBXX3E	FDCBXX3E

	S	Z	H	P/V	N	C	
RR/RL	*	*	0	*	0	*	Rotiere Reg rechts/links durch Carry
RRC/RLC	*	*	0	*	0	*	Rotiere Reg rechts/links
SRA/SLA	*	*	0	*	0	*	Shift Reg rechts/links arithmetisch
SRL	*	*	0	*	0	*	Shift Reg rechts/links logisch

		S	Z	H	P/V	N	C	
RRCA	0F	-	-	0	-	0	*	Rotiere Akku rechts
RLCA	07	-	-	0	-	0	*	Rotiere Akku links
RRA	1F	-	-	0	-	0	*	Rotiere Akku rechts durch Carry
RLA	17	-	-	0	-	0	*	Rotiere Akku links durch Carry
RLD(HL)	ED6F	*	*	0	*	0	-	Rotiere Ziffer links zwischen Akku und (HL)
RRD(HL)	ED67	*	*	0	*	0	-	Rotiere Ziffer rechts zwischen Akku und (HL)

Einzelbitbefehle

	B	C	D	E	H	L	(HL)	A	(IX+d)	(IY+d)
BIT 0	CB40	CB41	CB42	CB43	CB44	CB45	CB46	CB47	DDCBXX46	FDCBXX46
BIT 1	CB48	CB49	CB4A	CB4B	CB4C	CB4D	CB4E	CB4F	DDCBXX4E	FDCBXX4E
BIT 2	CB50	CB51	CB52	CB53	CB54	CB55	CB56	CB57	DDCBXX56	FDCBXX56
BIT 3	CB58	CB59	CB5A	CB5B	CB5C	CB5D	CB5E	CB5F	DDCBXX5E	FDCBXX5E
BIT 4	CB60	CB61	CB62	CB63	CB64	CB65	CB66	CB67	DDCBXX66	FDCBXX66
BIT 5	CB68	CB69	CB6A	CB6B	CB6C	CB6D	CB6E	CB6F	DDCBXX6E	FDCBXX6E
BIT 6	CB70	CB71	CB72	CB73	CB74	CB75	CB76	CB77	DDCBXX76	FDCBXX76
BIT 7	CB78	CB79	CB7A	CB7B	CB7C	CB7D	CB7E	CB7F	DDCBXX7E	FDCBXX7E
RES 0	CB80	CB81	CB82	CB83	CB84	CB85	CB86	CB87	DDCBXX86	FDCBXX86
RES 1	CB88	CB89	CB8A	CB8B	CB8C	CB8D	CB8E	CB8F	DDCBXX8E	FDCBXX8E
RES 2	CB90	CB91	CB92	CB93	CB94	CB95	CB96	CB97	DDCBXX96	FDCBXX96
RES 3	CB98	CB99	CB9A	CB9B	CB9C	CB9D	CB9E	CB9F	DDCBXX9E	FDCBXX9E
RES 4	CBA0	CBA1	CBA2	CBA3	CBA4	CBA5	CBA6	CBA7	DDCBXXA6	FDCBXXA6
RES 5	CBA8	CBA9	CBAA	CBAB	CBAC	CBAD	CBAE	CBAF	DDCBXXAE	FDCBXXAE
RES 6	CBB0	CBB1	CBB2	CBB3	CBB4	CBB5	CBB6	CBB7	DDCBXXB6	FDCBXXB6
RES 7	CBB8	CBB9	CBBA	CBBB	CBBC	CBBD	CBBE	CBBF	DDCBXXBE	FDCBXXBE
SET 0	CBC0	CBC1	CBC2	CBC3	CBC4	CBC5	CBC6	CBC7	DDCBXXC6	FDCBXXC6
SET 1	CBC8	CBC9	CBCA	CBCB	CBCC	CBCD	CBCE	CBCF	DDCBXXCE	FDCBXXCE
SET 2	CBD0	CBD1	CBD2	CBD3	CBD4	CBD5	CBD6	CBD7	DDCBXXD6	FDCBXXD6
SET 3	CBD8	CBD9	CBDA	CBDB	CBDC	CBDD	CBDE	CBDF	DDCBXXDE	FDCBXXDE
SET 4	CBE0	CBE1	CBE2	CBE3	CBE4	CBE5	CBE6	CBE7	DDCBXXE6	FDCBXXE6
SET 5	CBE8	CBE9	CBEA	CBEB	CBEC	CBED	CBEE	CBEF	DDCBXXEE	FDCBXXEE
SET 6	CBF0	CBF1	CBF2	CBF3	CBF4	CBF5	CBF6	CBF7	DDCBXXF6	FDCBXXF6
SET 7	CBF8	CBF9	CBFA	CBFB	CBFC	CBFD	CBFE	CBFF	DDCBXXFE	FDCBXXFE

Flagbeeinfussung:

	S	Z	H	P/V	N	C
BIT	?	*	1	?	0	-
SET	-	-	-	-	-	-
RES	-	-	-	-	-	-

Flag-Register

Bit	7	6	5	4	3	2	1	0
	S	Z	X	H	X	P/V	N	C

	gesetzt	nicht gesetzt	wird gesetzt bei
C Carry-Flag	C	NC	Uebertrag von Bit 7
N Add-/Subtract-Flag			Subtraktionen
P/V Parity-/Overflow-Flag	PE	PO	gerader Paritaet
H Half-Carry-Flag			Uebertrag von Bit 3
Z Zero-Flag	Z	NZ	Ergebnis 0
S Sign-Flag	M	P	neg. Ergebnis
X nicht verwendet			

Beeinflussung:
1 gesetzt
0 zurueckgesetzt
* abhaengig vom Ergebnis einer Operation
- nicht beeinflusst
? unbestimmt

8-bit Arithmetische und Logische Befehle

	B	C	D	E	H	L	(HL)	A	n	(IX+d)	(IY+d)	S	Z	H	P/V	N	C
ADD .	80	81	82	83	84	85	86	87	C6XX	DD86XX	FD86XX	*	*	*	*	0	*
ADC .	88	89	8A	8B	8C	8D	8E	8F	CEXX	DD8EXX	FD8EXX	*	*	*	*	0	*
SUB .	90	91	92	93	94	95	96	97	D6XX	DD96XX	FD96XX	*	*	*	*	1	*
SBC .	98	99	9A	9B	9C	9D	9E	9F	DEXX	DD9EXX	FD9EXX	*	*	*	*	1	*
AND .	A0	A1	A2	A3	A4	A5	A6	A7	E6XX	DDA6XX	FDA6XX	*	*	1	*	0	0
XOR .	A8	A9	AA	AB	AC	AD	AE	AF	EEXX	DDAEXX	FDAEXX	*	*	1	*	0	0
OR .	B0	B1	B2	B3	B4	B5	B6	B7	F6XX	DDB6XX	FDB6XX	*	*	1	*	0	0
CP .	B8	B9	BA	BB	BC	BD	BE	BF	FEXX	DDBEXX	FDBEXX	*	*	*	*	1	*
INC .	04	0C	14	1C	24	2C	34	3C		DD34XX	FD34XX	*	*	*	*	0	-
DEC .	05	0D	15	1D	25	2D	35	3D		DD35XX	FD35XX	*	*	*	*	1	-

		S	Z	H	P/V	N	C	
DAA	27	*	*	*	*	-	*	BCD-Korrektur im Akku
CPL	2F	-	-	1	-	1	-	Komplementiere Akku (1er-Komplement)
NEG	ED44	*	*	*	*	1	*	Komplementiere Akku (2er-Komplement)

16-bit Arithmetische und Logische Befehle

	BC	DE	HL	SP	IX	IY	S	Z	H	P/V	N	C
INC .	03	13	23	33	DD23	FD23	-	-	-	-	-	-
DEC .	0B	1B	2B	3B	DD2B	FD2B	-	-	-	-	-	-
ADD HL,.	09	19	29	39			-	-	*	-	0	*
ADC HL,.	ED4A	ED5A	ED6A	ED7A			*	*	*	*	0	*
SBC HL,.	ED42	ED52	ED62	ED72			*	*	*	*	1	*
ADD IX,.	DD09	DD19		DD39	DD29		-	-	*	-	0	*
ADD IY,.	FD09	FD19		FD39		FD29	-	-	*	-	0	*

Rotations- und Schiebebefehle

	B	C	D	E	H	L	(HL)	A	(IX+d)	(IY+d)
RR	CB18	CB19	CB1A	CB1B	CB1C	CB1D	CB1E	CB1F	DDCBXX1E	FDCBXX1E
RL	CB10	CB11	CB12	CB13	CB14	CB15	CB16	CB17	DDCBXX16	FDCBXX16
RRC	CB08	CB09	CB0A	CB0B	CB0C	CB0D	CB0E	CB0F	DDCBXX0E	FDCBXX0E
RLC	CB00	CB01	CB02	CB03	CB04	CB05	CB06	CB07	DDCBXX06	FDCBXX06
SRA	CB28	CB29	CB2A	CB2B	CB2C	CB2D	CB2E	CB2F	DDCBXX2E	FDCBXX2E
SLA	CB20	CB21	CB22	CB23	CB24	CB25	CB26	CB27	DDCBXX26	FDCBXX26
SRL	CB38	CB39	CB3A	CB3B	CB3C	CB3D	CB3E	CB3F	DDCBXX3E	FDCBXX3E

	S	Z	H	P/V	N	C	
RR/RL	*	*	0	*	0	*	Rotiere Reg rechts/links durch Carry
RRC/RLC	*	*	0	*	0	*	Rotiere Reg rechts/links
SRA/SLA	*	*	0	*	0	*	Shift Reg rechts/links arithmetisch
SRL	*	*	0	*	0	*	Shift Reg rechts/links logisch

		S	Z	H	P/V	N	C	
RRCA	0F	-	-	0	-	0	*	Rotiere Akku rechts
RLCA	07	-	-	0	-	0	*	Rotiere Akku links
RRA	1F	-	-	0	-	0	*	Rotiere Akku rechts durch Carry
RLA	17	-	-	0	-	0	*	Rotiere Akku links durch Carry
RLD(HL)	ED6F	*	*	0	*	0	-	Rotiere Ziffer links zwischen Akku und (HL)
RRD(HL)	ED67	*	*	0	*	0	-	Rotiere Ziffer rechts zwischen Akku und (HL)

Einzelbitbefehle

	B	C	D	E	H	L	(HL)	A	(IX+d)	(IY+d)
BIT 0	CB40	CB41	CB42	CB43	CB44	CB45	CB46	CB47	DDCBXX46	FDCBXX46
BIT 1	CB48	CB49	CB4A	CB4B	CB4C	CB4D	CB4E	CB4F	DDCBXX4E	FDCBXX4E
BIT 2	CB50	CB51	CB52	CB53	CB54	CB55	CB56	CB57	DDCBXX56	FDCBXX56
BIT 3	CB58	CB59	CB5A	CB5B	CB5C	CB5D	CB5E	CB5F	DDCBXX5E	FDCBXX5E
BIT 4	CB60	CB61	CB62	CB63	CB64	CB65	CB66	CB67	DDCBXX66	FDCBXX66
BIT 5	CB68	CB69	CB6A	CB6B	CB6C	CB6D	CB6E	CB6F	DDCBXX6E	FDCBXX6E
BIT 6	CB70	CB71	CB72	CB73	CB74	CB75	CB76	CB77	DDCBXX76	FDCBXX76
BIT 7	CB78	CB79	CB7A	CB7B	CB7C	CB7D	CB7E	CB7F	DDCBXX7E	FDCBXX7E
RES 0	CB80	CB81	CB82	CB83	CB84	CB85	CB86	CB87	DDCBXX86	FDCBXX86
RES 1	CB88	CB89	CB8A	CB8B	CB8C	CB8D	CB8E	CB8F	DDCBXX8E	FDCBXX8E
RES 2	CB90	CB91	CB92	CB93	CB94	CB95	CB96	CB97	DDCBXX96	FDCBXX96
RES 3	CB98	CB99	CB9A	CB9B	CB9C	CB9D	CB9E	CB9F	DDCBXX9E	FDCBXX9E
RES 4	CBA0	CBA1	CBA2	CBA3	CBA4	CBA5	CBA6	CBA7	DDCBXXA6	FDCBXXA6
RES 5	CBA8	CBA9	CBAA	CBAB	CBAC	CBAD	CBAE	CBAF	DDCBXXAE	FDCBXXAE
RES 6	CBB0	CBB1	CBB2	CBB3	CBB4	CBB5	CBB6	CBB7	DDCBXXB6	FDCBXXB6
RES 7	CBB8	CBB9	CBBA	CBBB	CBBC	CBBD	CBBE	CBBF	DDCBXXBE	FDCBXXBE
SET 0	CBC0	CBC1	CBC2	CBC3	CBC4	CBC5	CBC6	CBC7	DDCBXXC6	FDCBXXC6
SET 1	CBC8	CBC9	CBCA	CBCB	CBCC	CBCD	CBCE	CBCF	DDCBXXCE	FDCBXXCE
SET 2	CBD0	CBD1	CBD2	CBD3	CBD4	CBD5	CBD6	CBD7	DDCBXXD6	FDCBXXD6
SET 3	CBD8	CBD9	CBDA	CBDB	CBDC	CBDD	CBDE	CBDF	DDCBXXDE	FDCBXXDE
SET 4	CBE0	CBE1	CBE2	CBE3	CBE4	CBE5	CBE6	CBE7	DDCBXXE6	FDCBXXE6
SET 5	CBE8	CBE9	CBEA	CBEB	CBEC	CBED	CBEE	CBEF	DDCBXXEE	FDCBXXEE
SET 6	CBF0	CBF1	CBF2	CBF3	CBF4	CBF5	CBF6	CBF7	DDCBXXF6	FDCBXXF6
SET 7	CBF8	CBF9	CBFA	CBFB	CBFC	CBFD	CBFE	CBFF	DDCBXXFE	FDCBXXFE

Flagbeeinfussung:

	S	Z	H	P/V	N	C
BIT	?	*	1	?	0	-
SET	-	-	-	-	-	-
RES	-	-	-	-	-	-

Flag-Register

Bit	7	6	5	4	3	2	1	0
	S	Z	X	H	X	P/V	N	C

	gesetzt	nicht gesetzt	wird gesetzt bei
C Carry-Flag	C	NC	Uebertrag von Bit 7
N Add-/Subtract-Flag			Subtraktionen
P/V Parity-/Overflow-Flag	PE	PO	gerader Paritaet
H Half-Carry-Flag			Uebertrag von Bit 3
Z Zero-Flag	Z	NZ	Ergebnis 0
S Sign-Flag	M	P	neg. Ergebnis
X nicht verwendet			

Beeinflussung:
1 gesetzt
0 zurueckgesetzt
* abhaengig vom Ergebnis einer Operation
- nicht beeinflusst
? unbestimmt


Der NDR-Klein Computer
PROMER
Komplette Preisliste und weitere Informationen in der Heftmitte.
Nutzen Sie die anhängende Bestellkarte!
Im Lauf der Schulfernseh-Sendereihe „Mikroelektronik“ werden in kleinen Schritten verschiedene Mikrocomputer aufgebaut.
Von der Ampelsteuerung über Roboter und einen in Maschinensprache programmierbaren 8-Bit-Rechner (Z80) bis hin zum nebenstehenden 16-Bit-Gerät.
Die hochauflösende Grafik-Bildschirmausgabe ermöglicht viele Anwendungen im Bereich Steuerung und Regelung.
MINIBUS
SBC 2
IOE
POW 5V
Schon in der kleinsten Ausbaustufe kann der NDR-Klein Computer den Fischertechnik-Roboter steuern.
Auf der Z80-Prozessorkarte SBC2 ist auch der RAM-Speicher und das Steuerprogramm in Eprom untergebracht.
Die IOE-Karte für universelle Ein/Ausgabe wurde hier um Treibertransistoren für den Roboter erweitert.
Die POW 5V liefert die stabilisierte Spannung für den Computer.
ELEKTRONIKLADEN 4930 DETMOLD 18 ☎ 05232/8171
Verkaufsstelle München, Schulstr. 28, 8000 München 19, Tel. 089/1679499